Университет ИТМО

Факультет ПИиКТ

Лабораторная работа №3 по дисциплине
“Системы искусственного интеллекта”

(3ий курс бакалавриата ФПИиКТ)

Студент:

Иванов Евгений Дмитриевич

Группа P33111

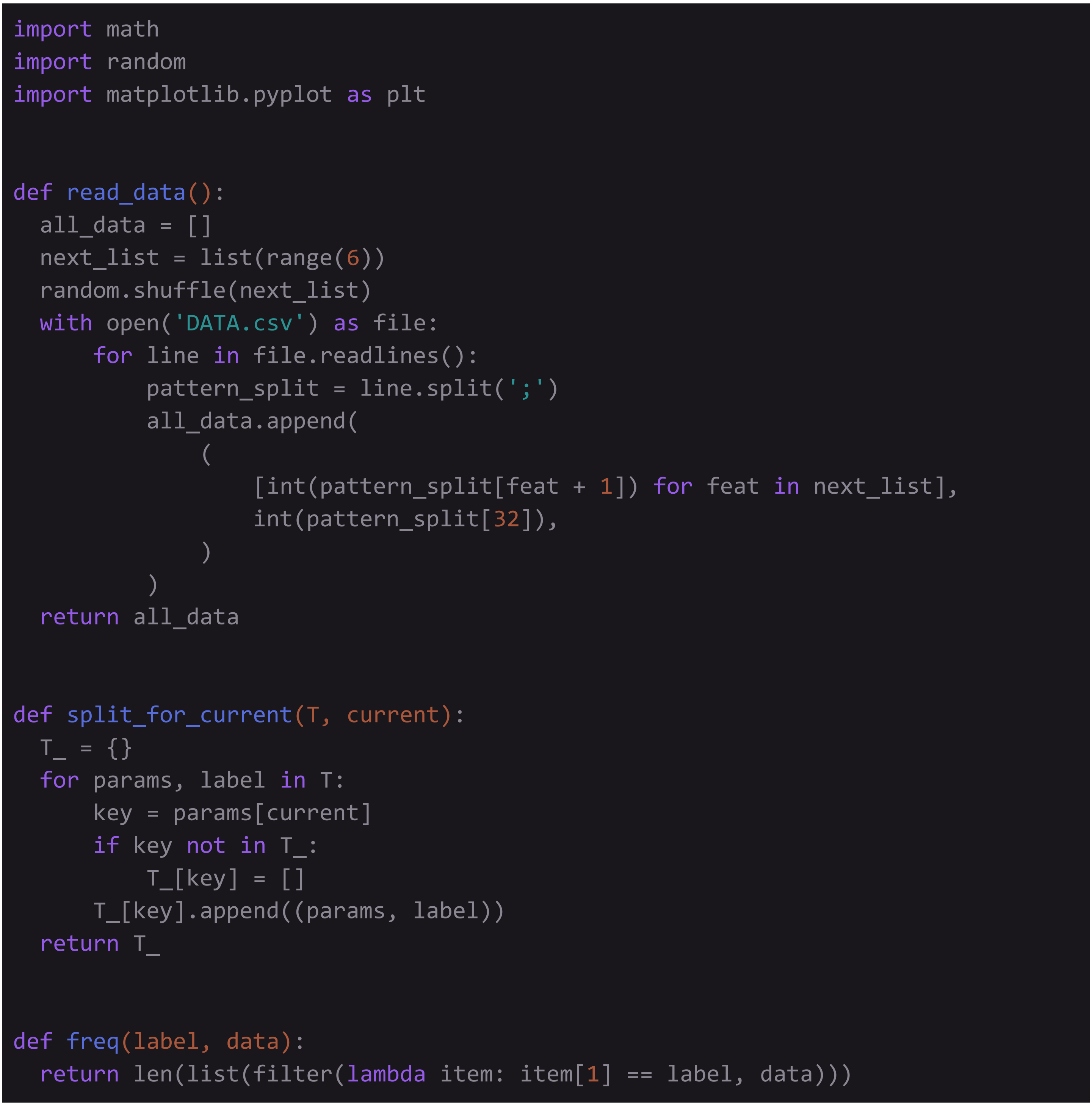
Преподаватель:

Авдюшина Анна Евгеньевна

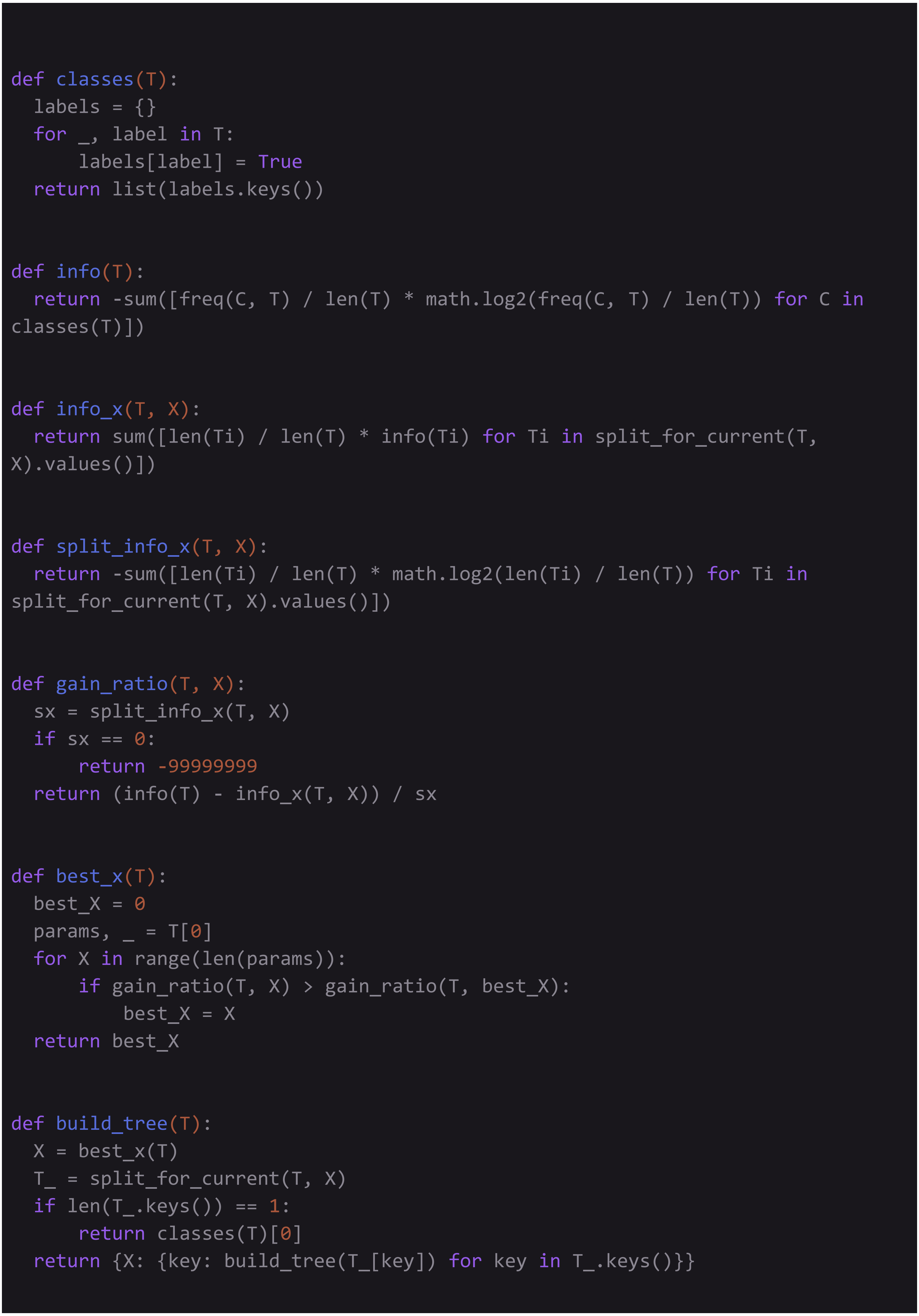
Задание:

1. Датасет с данными про оценки студентов инженерного и педагогического факультетов
2. Отобрать случайным образом sqrt(n) признаков
3. Реализовать без использования сторонних библиотек построение дерева решений (numpy и pandas использовать можно)
4. Провести оценку реализованного алгоритма с использованием Accuracy, precision и recall
5. Построить AUC-ROC и AUC-PR

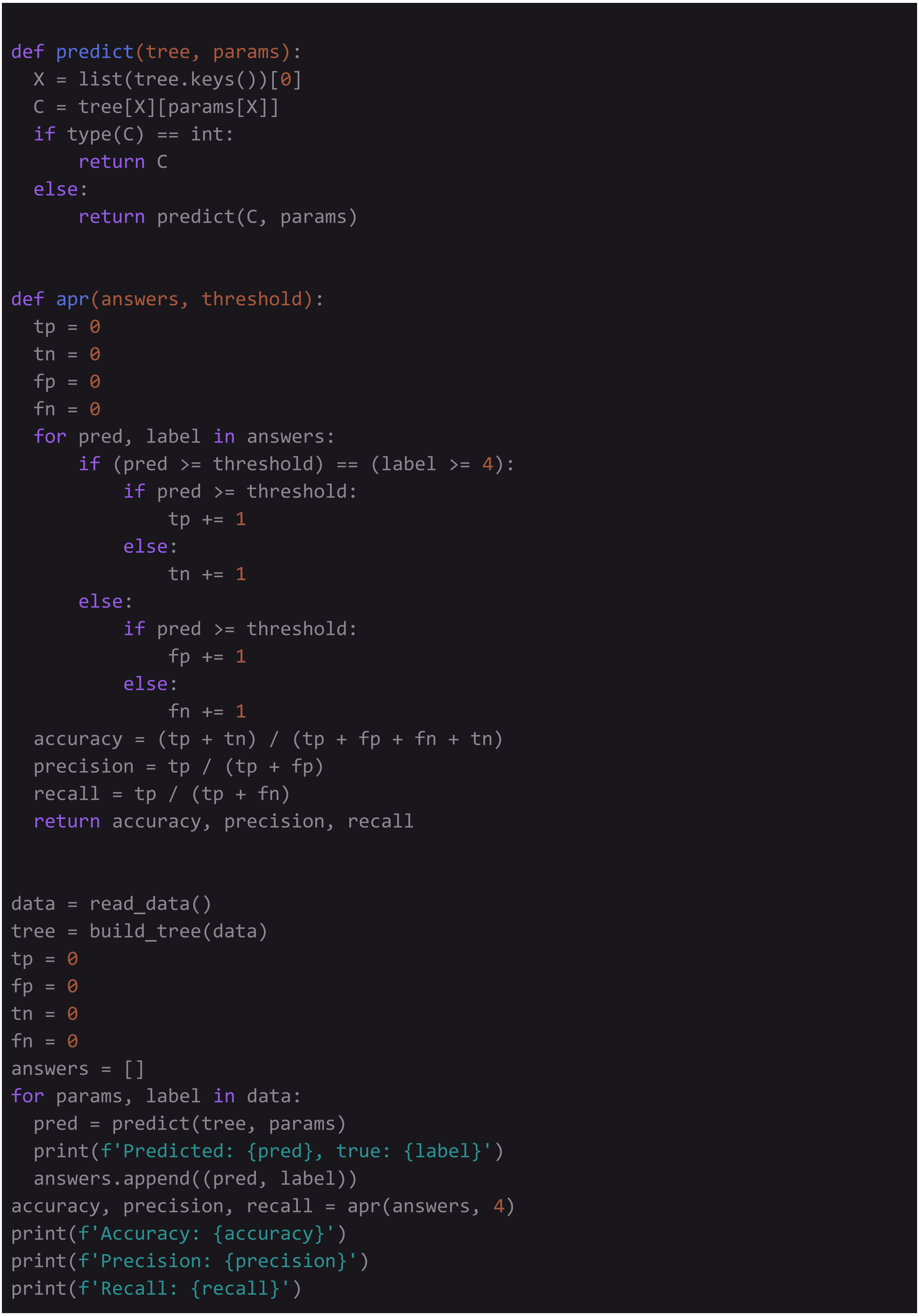
Код программы:



```
import math
import random
import matplotlib.pyplot as plt


def read_data():
  all_data = []
  next_list = list(range(6))
  random.shuffle(next_list)
  with open('DATA.csv') as file:
      for line in file.readlines():
          pattern_split = line.split(';')
          all_data.append(
              (
                  [int(pattern_split[feat + 1]) for feat in next_list],
                  int(pattern_split[32]),
              )
          )
  return all_data


def split_for_current(T, current):
  T_ = {}
  for params, label in T:
      key = params[current]
      if key not in T_:
          T_[key] = []
      T_[key].append((params, label))
  return T_


def freq(label, data):
  return len(list(filter(lambda item: item[1] == label, data)))
```



```
def classes(T):
  labels = {}
  for _, label in T:
      labels[label] = True
  return list(labels.keys())


def info(T):
  return -sum([freq(C, T) / len(T) * math.log2(freq(C, T) / len(T)) for C in
classes(T)])


def info_x(T, X):
  return sum([len(Ti) / len(T) * info(Ti) for Ti in split_for_current(T,
X).values()])


def split_info_x(T, X):
  return -sum([len(Ti) / len(T) * math.log2(len(Ti) / len(T)) for Ti in
split_for_current(T, X).values()])


def gain_ratio(T, X):
  sx = split_info_x(T, X)
  if sx == 0:
      return -99999999
  return (info(T) - info_x(T, X)) / sx


def best_x(T):
  best_X = 0
  params, _ = T[0]
  for X in range(len(params)):
      if gain_ratio(T, X) > gain_ratio(T, best_X):
          best_X = X
  return best_X


def build_tree(T):
  X = best_x(T)
  T_ = split_for_current(T, X)
  if len(T_.keys()) == 1:
      return classes(T)[0]
  return {X: {key: build_tree(T_[key]) for key in T_.keys()}}
```



```
def predict(tree, params):
  X = list(tree.keys())[0]
  C = tree[X][params[X]]
  if type(C) == int:
      return C
  else:
      return predict(C, params)


def apr(answers, threshold):
  tp = 0
  tn = 0
  fp = 0
  fn = 0
  for pred, label in answers:
      if (pred >= threshold) == (label >= 4):
          if pred >= threshold:
              tp += 1
          else:
              tn += 1
      else:
          if pred >= threshold:
              fp += 1
          else:
              fn += 1
  accuracy = (tp + tn) / (tp + fp + fn + tn)
  precision = tp / (tp + fp)
  recall = tp / (tp + fn)
  return accuracy, precision, recall


data = read_data()
tree = build_tree(data)
tp = 0
fp = 0
tn = 0
fn = 0
answers = []
for params, label in data:
  pred = predict(tree, params)
  print(f'Predicted: {pred}, true: {label}')
  answers.append((pred, label))
accuracy, precision, recall = apr(answers, 4)
print(f'Accuracy: {accuracy}')
print(f'Precision: {precision}')
print(f'Recall: {recall}')
```

```
answers.sort(key=lambda item: -item[0])
n = len(list(filter(lambda item: item[1] >= 4, data)))
m = len(list(filter(lambda item: item[1] < 4, data)))
x = [0]
y = [0]
for i in range(0, len(answers)):
  if answers[i][1] >= 4:
      x.append(x[-1])
      y.append(y[-1] + 1)
  else:
      x.append(x[-1] + 1)
      y.append(y[-1])
plt.plot(x, y)
plt.title("AUC-ROC")
plt.show()

x = []
y = []
prev = None
for threshold in range(1, 7):
  _, precision, recall = apr(answers, threshold)
  if prev is not None:
      x += [prev]
      y += [precision]
  x += [recall]
  y += [precision]
  prev = recall
plt.plot(x, y)
plt.title("AUC-PR")
plt.show()
```

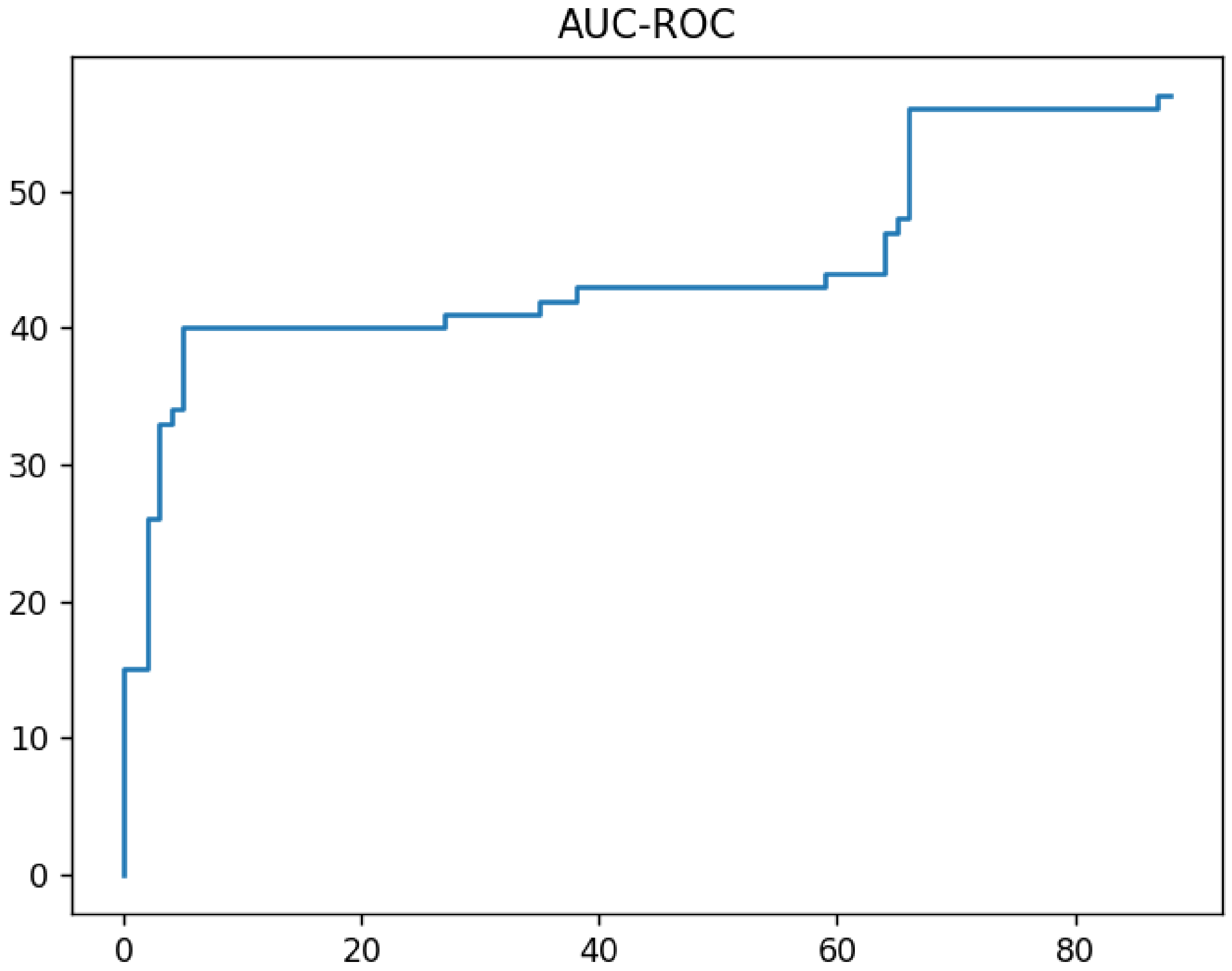
Результат работы программы:

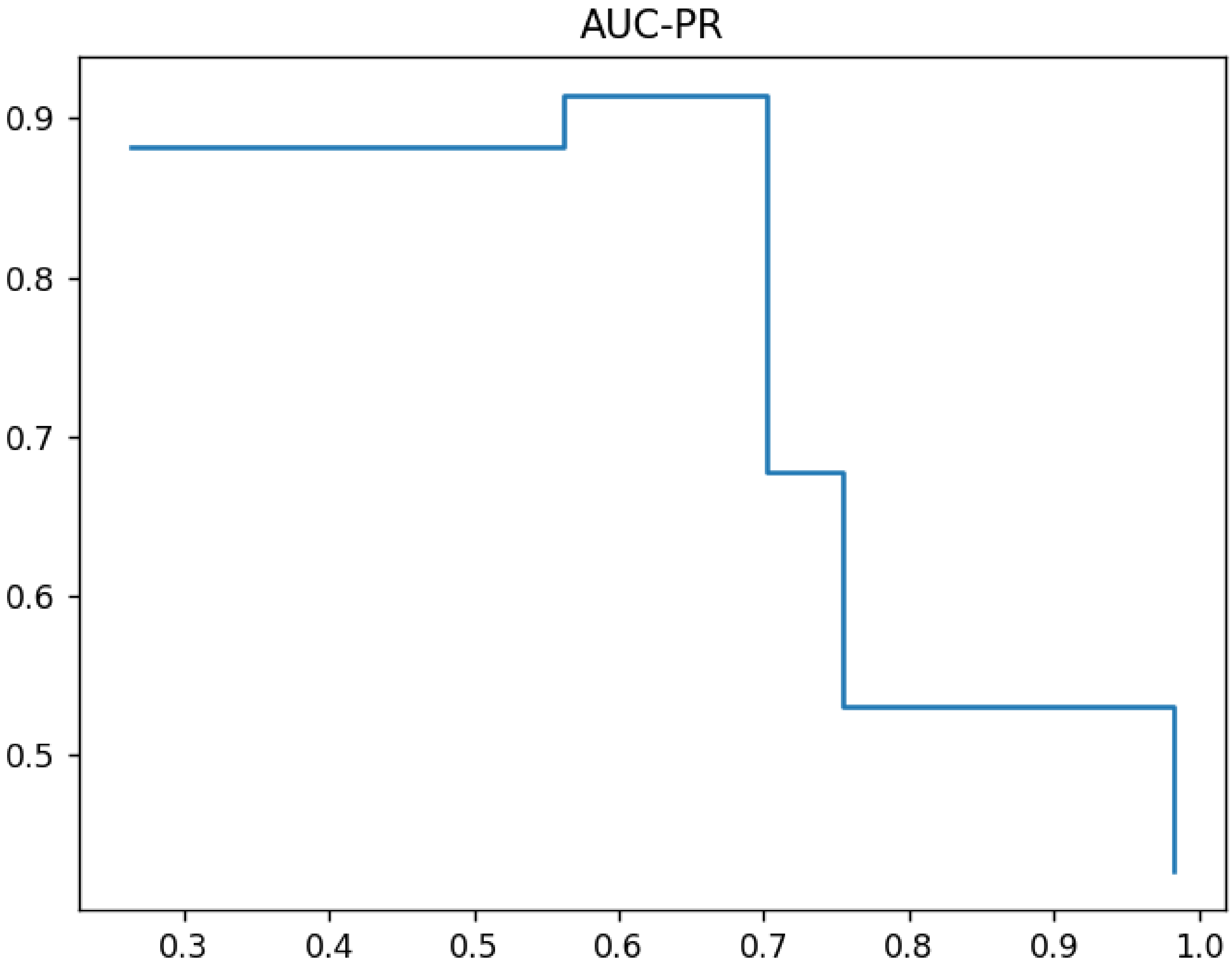
```
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 2
Predicted: 5, true: 5
Predicted: 2, true: 2
Predicted: 5, true: 5
Predicted: 0, true: 0
Predicted: 2, true: 2
Predicted: 0, true: 0
Predicted: 0, true: 0
Predicted: 1, true: 1
```

```
Predicted: 2, true: 2
Predicted: 1, true: 2
Predicted: 1, true: 1
Predicted: 1, true: 2
Predicted: 2, true: 2
Predicted: 3, true: 3
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 3, true: 3
Predicted: 1, true: 1
Predicted: 1, true: 2
Predicted: 1, true: 3
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 3, true: 3
Predicted: 5, true: 5
Predicted: 5, true: 5
Predicted: 3, true: 3
Predicted: 0, true: 1
Predicted: 2, true: 2
Predicted: 3, true: 2
Predicted: 1, true: 1
Predicted: 2, true: 2
Predicted: 1, true: 1
Predicted: 2, true: 2
Predicted: 0, true: 1
Predicted: 3, true: 1
Predicted: 3, true: 1
Predicted: 1, true: 1
Predicted: 4, true: 4
Predicted: 1, true: 1
Predicted: 4, true: 3
Predicted: 1, true: 5
Predicted: 1, true: 3
Predicted: 1, true: 1
Predicted: 1, true: 2
Predicted: 1, true: 1
Predicted: 4, true: 4
Predicted: 2, true: 1
Predicted: 5, true: 5
Predicted: 1, true: 3
Predicted: 3, true: 3
Predicted: 1, true: 5
Predicted: 1, true: 4
Predicted: 3, true: 3
Predicted: 1, true: 5
Predicted: 0, true: 2
Predicted: 5, true: 5
Predicted: 1, true: 3
```

```
Predicted: 5, true: 5
Predicted: 3, true: 3
Predicted: 4, true: 2
Predicted: 1, true: 5
Predicted: 1, true: 1
Predicted: 5, true: 5
Predicted: 5, true: 5
Predicted: 7, true: 7
Predicted: 6, true: 6
Predicted: 1, true: 6
Predicted: 5, true: 6
Predicted: 5, true: 7
Predicted: 7, true: 7
Predicted: 4, true: 4
Predicted: 7, true: 7
Predicted: 4, true: 4
Predicted: 5, true: 3
Predicted: 2, true: 4
Predicted: 3, true: 3
Predicted: 7, true: 7
Predicted: 1, true: 7
Predicted: 7, true: 7
Predicted: 1, true: 4
Predicted: 5, true: 5
Predicted: 1, true: 6
Predicted: 7, true: 6
Predicted: 1, true: 6
Predicted: 5, true: 6
Predicted: 5, true: 6
Predicted: 4, true: 7
Predicted: 4, true: 4
Predicted: 1, true: 6
Predicted: 5, true: 5
Predicted: 7, true: 7
Predicted: 6, true: 6
Predicted: 7, true: 7
Predicted: 1, true: 7
Predicted: 5, true: 6
Predicted: 5, true: 7
Predicted: 4, true: 7
Predicted: 1, true: 7
Predicted: 3, true: 3
Predicted: 7, true: 7
Predicted: 7, true: 7
Predicted: 6, true: 6
Predicted: 6, true: 6
Predicted: 7, true: 7
Predicted: 1, true: 2
Predicted: 2, true: 2
```

```
Predicted: 2, true: 2
Predicted: 1, true: 1
Predicted: 2, true: 2
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 1, true: 1
Predicted: 2, true: 2
Predicted: 1, true: 1
Predicted: 0, true: 0
Predicted: 1, true: 2
Predicted: 1, true: 1
Predicted: 6, true: 3
Predicted: 2, true: 2
Predicted: 0, true: 3
Predicted: 2, true: 1
Predicted: 0, true: 0
Predicted: 2, true: 3
Predicted: 2, true: 1
Predicted: 2, true: 4
Predicted: 2, true: 3
Predicted: 3, true: 3
Predicted: 0, true: 1
Predicted: 0, true: 2
Predicted: 2, true: 0
Predicted: 2, true: 2
Predicted: 6, true: 0
Predicted: 0, true: 0
Predicted: 0, true: 5
Predicted: 2, true: 5
Predicted: 0, true: 1
Predicted: 4, true: 4
Predicted: 3, true: 3
Accuracy: 0.8482758620689655
Precision: 0.8888888888888888
Recall: 0.7017543859649122
```


AUC-ROC
0
10
20
30
40
50
0
20
40
60
80



Вывод: Научился работать с разбиением и классификацией информации при помощи алгоритма C4.5. Разобрался с такими графиками и характеристиками как AUC-ROC, AUC-PR.