Введение в Python

Гейне М.А.

11.09.2024

Что такое Python?

Python - это высокоуровневый язык программирования общего назначения, известный своей простотой, читабельностью и универсальностью. Он широко используется в различных областях, включая веб-разработку, науку о данных, машинное обучение, автоматизацию и научные вычисления.

История Python

Python 0.9.0

- Февраль 1991, Python 0.9.0 в общем доступе. Разрабатывался Гвидо ван Россумом в качестве нового учебного языка на замену ABC.
- Классы с наследованием, исключения, функции, `list`, `dict`, `str` и прочие базовые типы
- Система модулей
- В 1994 появился основной форум Python -> цель на открытость системы

Январь 1994, Python 1.0

- `lambda`, `map`, `filter`, `reduce`. Позаимствованы из Lisp сообществом.
- 1.4: `keyword arguments`, комплексные числа, "сокрытие данных путём искажения имён"
- Computer Programming for Everybody (CP4E)
- BeOpen и GPL совместимая лицензия

Октябрь 2000, Python 2.0

- List comprehensions (списочное выражение), garbage collector - из ФП
- Python Software Foundation License - собственная открытая лицензия, которая разрешает в том числе приобритение прав на производные работы
- Декабрь 2001, Python 2.2, унификация типов
- Сентябрь 2006, Python 2.5, захват ресурсов выражением `with`. RAII.

Подготовка к Python 3

- Python 2.6 & 2.7 выходили вместе с 3.0 & 3.1, содержали часть фич из старших релизов, а также предупреждения об удалении фич в новых версиях. В ноябре 2014 объявлено о прекращении поддержки версии 2.7 в 2020 году. Последний релиз, 2.7.18, состоялся 20 апреля 2020 года

3 декабря 2008, Python 3.0

- Проектировался, чтобы исправить фундаментальные недостатки языка -> обратно несовместимые изменения -> цифра 3 (см. семантическое версионирование)
- "reduce feature duplication by removing old ways of doing things" - сокращаем повторяющиеся фичи за счёт устранения старых способов работы
- "There should be one— and preferably only one — obvious way to do it"

Проблемы совместимости

- Смена версий не прошла безболезненно. Были средства автоматизации (например `2to3`), но они справлялись с задачей лишь частично
- Изначальная схема совместимости: держите всю кодовую базу на Python 2, а для Python 3 делайте доп версию с помощью `2to3`, и изменения туда лучше не вносить. С 2012 года было решено держать только одну кодовую версию и для 2, и для 3 версии, а совместимости добиваться подключением модулей совместимости

Основные фичи Python 3:

- `print` стал функцией, а не выражением -> легко заменить его поведение
- Замена функции `input`, которая раньше не просто возвращала строку, а вычисляла входное выражение
- Убран `reduce` из-за меньшей читабельности, чем у циклов
- Добавлены аннотации функций

Основные фичи Python 3:

- Типы str и Unicode были унифицированы
- Добавлены типы для последовательностей байтов
- Убраны особенности обратной совместимости
- Изменено поведение целочисленного деления. `9 / 2 = 4` vs `9 / 2 = 4.5`
- Полная поддержка Unicode в исходном коде

Другие крупные релизы после 3.0:

- 3.6: f-строки, подсказки типов, улучшения asyncio
- 3.7: Data classes, `async with`
- 3.9: `setdeafult()` и `popitem()` для словарей, улучшения в работе с таймзонами, изменение конвенций об именованиях встроенных модулей
- 3.10: Pattern matching, union operator `|`

Ключевые особенности Python

- **Удобство чтения:** Синтаксис Python разработан так, чтобы его было легко читать и понимать, что делает его отличным выбором для новичков.
- **Универсальность:** Python можно использовать для решения широкого круга задач, от простых скриптов до сложных приложений.
- **Большое сообщество:** Python имеет большое и активное сообщество, предоставляющее обширную документацию, библиотеки и поддержку

Ключевые особенности Python

- **Кроссплатформенная совместимость:** Python работает на различных операционных системах, включая Windows, macOS и Linux.
- **Динамическая типизация:** Python автоматически определяет тип данных переменных во время выполнения программы, что делает его более гибким по сравнению с языками со статической типизацией.
- **Интерпретация over компиляция**: Python исполняется интерпретатором, т.е. не имеет этапа компиляции программы.

Виды интерпретатора

1. **CPython.** Основной интерпретатор Python, наиболее распространённый и считающийся стандартом. Написан на, внезапно, C.
2. **PyPy.** JIT-компилятор для Python. Быстрее CPython, особенно в вычислительно сложных задачах, совместим с большинством библиотек
3. **Jython**. Python to Java bytecode
4. **IronPython**. Python to .NET
5. **Stackless Python**. CPython, но с отказом от стека вызовов C в пользу собственного

Применение Python

- Data Science and Machine Learning
  - NumPy, Pandas и Matplotlib - основные средства для интерпретации и визуализации данных
  - Sklearn, TensorFlow, Keras, PyTorch - ML фреймворки для построения и обучения моделей
- Scientific Computing
  - SciPy и SymPy - численные симуляции и математические исчисления
  - Matplotlib - визуализация научных данных

Применение Python

- Веб-разработка
  - Django, Flask, Pyramid - фреймворки для разработки веб приложений
  - BeautifulSoup и Scrappy - инструменты извлечения данных с веб-страниц
- Образование
  - Python - изначально учебный язык, и он до сих пор используется для обучения программированию

Применение Python

- Автоматизация
  - Простой скриптовый язык позволяет достаточно легко и быстро автоматизировать ручные рутинные задачи
- Системы управления
  - Home Assistant - система управления умным домом с открытым исходным кодом, работающая в основном на Python
  - Интеграции с умными устройствами пишутся на Python и доступны любому для редактирования

Пример анализа данных

0. Подготовка датасета


Объединённая таблица

1. Загрузка датасета

```
df = pd.read_excel('Dataset2.xlsx')

date_columns = ['OBGYN', 'AIDS', 'BirthDate']

for col in date_columns:
    df[col] = pd.to_datetime(df[col], format='%d.%m.%Y', errors='coerce')
```


Фрагмент очищенной таблицы

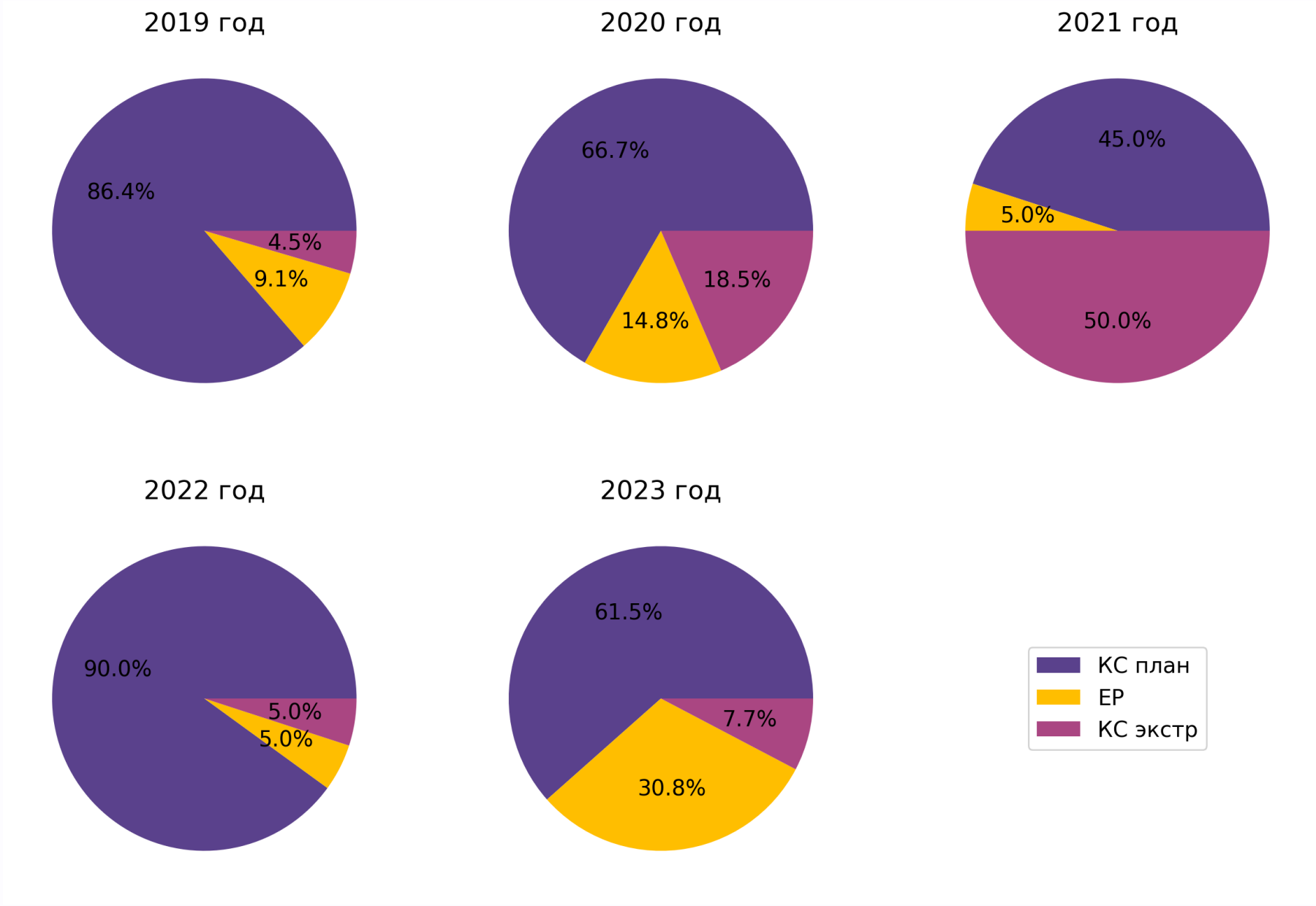
2. Трансформация данных

```
df['LengthDelta'] = pd.to_timedelta(df['Length'], unit='W')
df['PregnancyStart'] = df['BirthDate'] - df['LengthDelta']

df['OBGYN'] = df['OBGYN'].fillna(df['BirthDate'])
df['AIDS'] = df['AIDS'].fillna(df['BirthDate'])

missing_week = df[df['Week'] == '-']
missing_week['Week'] = missing_week['Length'] - (pd.to_timedelta((missing_week['BirthDate']
        - missing_week['OBGYN'])).dt.days // 7)
df.update(missing_week)
df['Week'] = df['Week'].astype(int)

# Initialize LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the 'BirthType' column
df['BirthType'] = label_encoder.fit_transform(df['BirthType'])
```

2. Трансформация данных


Обновлённый датасет

3. Визуализация данных



3. Визуализация данных

```
# Get yearly data
yearly_percentages = df.groupby('Year')['BirthType'].value_counts(normalize=True).sort_index() * 100

# Create a single figure with 5 subplots
fig, axs = plt.subplots(2, 3, figsize=(15, 10))

# Plot pie charts for each year
for i, (year, percentages) in enumerate(yearly_percentages.groupby(level=0)):
    # birth_types = [label[1] for label in percentages.index]
    wedges, texts, autotexts = axs[i//3][i%3].pie(percentages, autopct='%1.1f%%', colors=generate_colors(len(percentages)))
    axs[i//3][i%3].set_title(f'{year} год')
axs[-1, -1].legend(wedges, labels, loc='center', fontsize=14)
axs[-1, -1].axis('off')
plt.savefig('fig2.png', transparent=True)
plt.show()
```

4. Разбиение данных на выборки

```
X = df.drop(columns=['Dosage'])
y = df['Dosage']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=df['Dosage'])

print("X train (features):")
print(X_train.head())
print("\nX test (features):")
print(X_test.head())
print("\ny train (target variable):")
print(y_train.head())
print("\ny test (target variable):")
print(y_test.head())
```

4. Разбиение данных на выборки

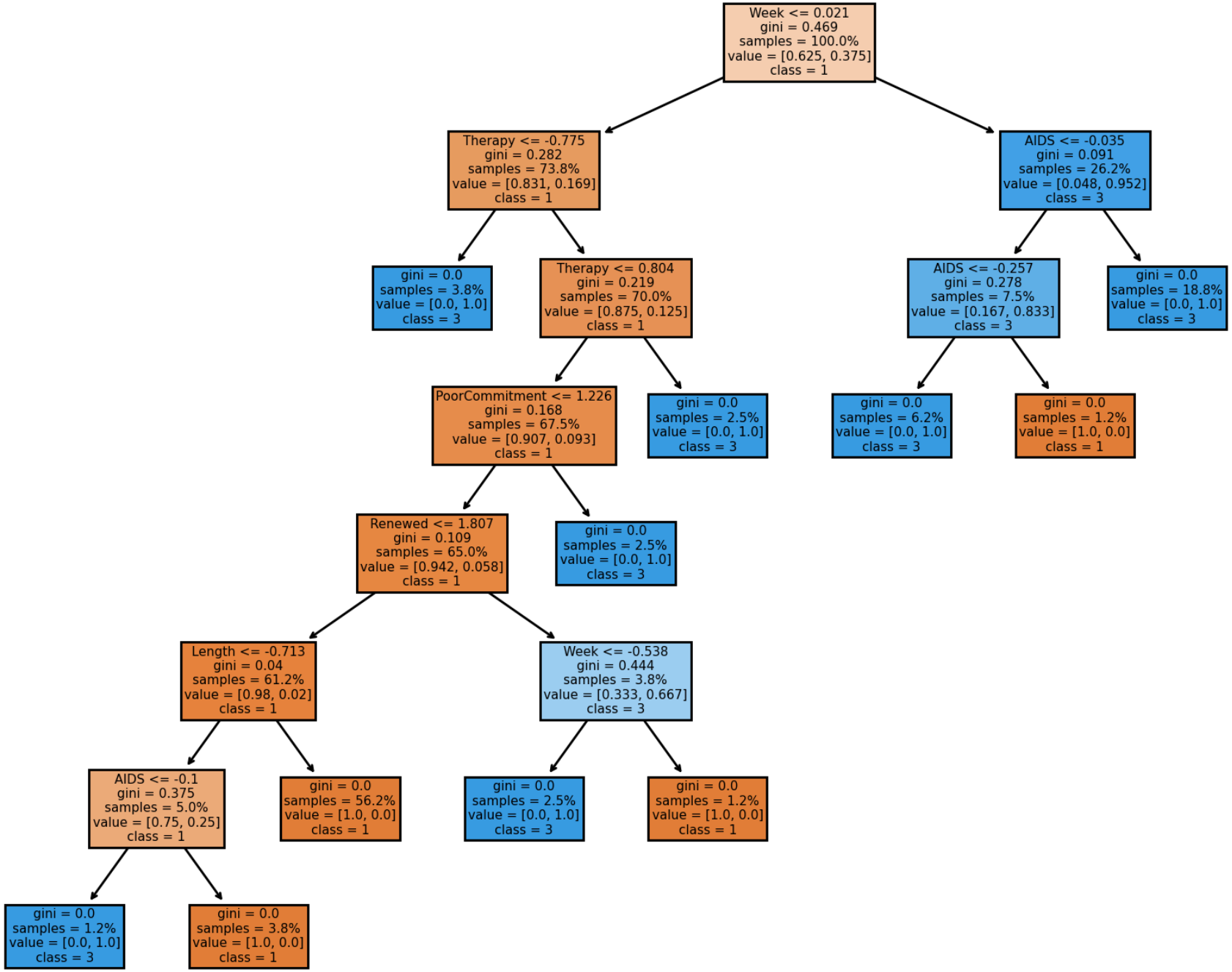
```
X train (features):
     Week  AIDS  BirthType  Therapy  Renewed  Status  Length  PoorCommitment  \
69      7 -5036          0        8      1.0     0.0      39             0.0
101    38   291          0       39      0.0     1.0      39             0.0
23      9  -536          0        0      0.0     0.0      38             0.0
51     11 -2024          2        0      0.0     0.0      35             0.0
19     30  -328          0        0      1.0     1.0      30             0.0

     SocialRisk  Migration  FSIN  Illness  Complications  Death
69          0.0        0.0   0.0      0.0            0.0    0.0
101         0.0        1.0   0.0      0.0            0.0    0.0
23          0.0        0.0   0.0      0.0            0.0    0.0
51          0.0        0.0   0.0      0.0            0.0    0.0
19          0.0        0.0   0.0      1.0            0.0    0.0
```

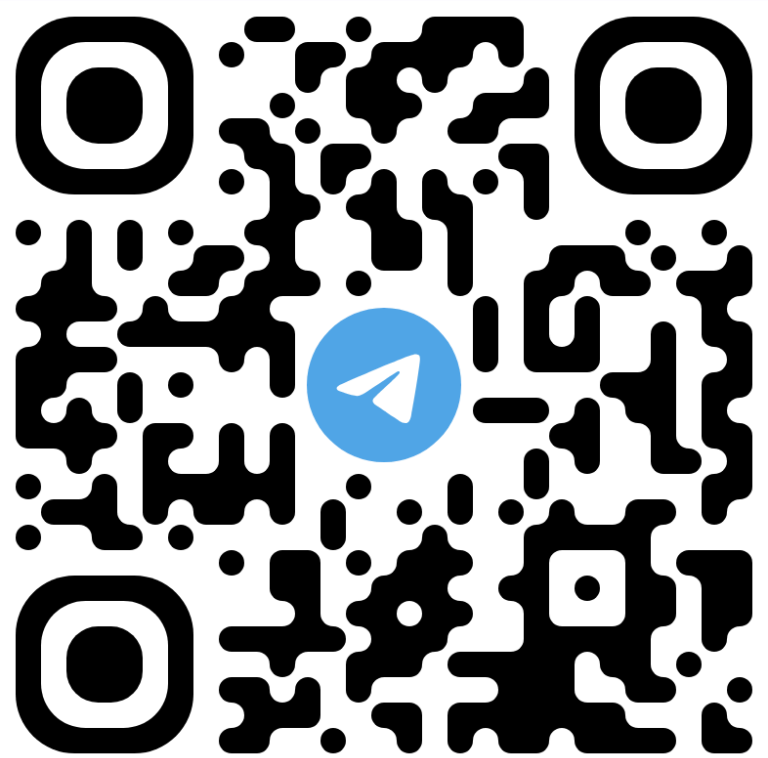
```
X test (features):
    Week  AIDS  BirthType  Therapy  Renewed  Status  Length  PoorCommitment  \
85    12 -4744          0        0      0.0     0.0      41             0.0
86     9 -3735          2        0      0.0     0.0      28             0.0
67    17   124          2       17      0.0     0.0      33             0.0
16    10 -3953          0        0      0.0     0.0      39             0.0
37     9 -3170          0        0      0.0     0.0      39             0.0

    SocialRisk  Migration  FSIN  Illness  Complications  Death
85         0.0        0.0   0.0      0.0            0.0    0.0
86         0.0        0.0   0.0      0.0            0.0    0.0
67         0.0        0.0   1.0      0.0            0.0    0.0
16         0.0        0.0   0.0      0.0            0.0    0.0
37         0.0        0.0   0.0      0.0            0.0    0.0
```

```
y train (target variable):
69     3.0
101    3.0
23     1.0
51     3.0
19     3.0
Name: Dosage, dtype: float64

y test (target variable):
85    1.0
86    1.0
67    3.0
16    1.0
37    1.0
Name: Dosage, dtype: float64
```

5. Обучение

```
# Create a StandardScaler object
scaler = StandardScaler()

# Fit the scaler on the training data
scaler.fit(X_train)

# Transform the training and testing data
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Instantiate the DecisionTreeClassifier with optional hyperparameters
tree_classifier_scaled = DecisionTreeClassifier(max_depth=50)

# Fit the model to the training data
tree_classifier_scaled.fit(X_train_scaled, y_train)

# Make predictions on the testing data
predictions_scaled = tree_classifier_scaled.predict(X_test_scaled)
```


Week <= 0.021
gini = 0.469
samples = 100.0%
value = [0.625, 0.375]
class = 1
Therapy <= -0.775
gini = 0.282
samples = 73.8%
value = [0.831, 0.169]
class = 1
AIDS <= -0.035
gini = 0.091
samples = 26.2%
value = [0.048, 0.952]
class = 3
gini = 0.0
samples = 3.8%
value = [0.0, 1.0]
class = 3
Therapy <= 0.804
gini = 0.219
samples = 70.0%
value = [0.875, 0.125]
class = 1
AIDS <= -0.257
gini = 0.278
samples = 7.5%
value = [0.167, 0.833]
class = 3
gini = 0.0
samples = 18.8%
value = [0.0, 1.0]
class = 3
PoorCommitment <= 1.226
gini = 0.168
samples = 67.5%
value = [0.907, 0.093]
class = 1
gini = 0.0
samples = 2.5%
value = [0.0, 1.0]
class = 3
gini = 0.0
samples = 6.2%
value = [0.0, 1.0]
class = 3
gini = 0.0
samples = 1.2%
value = [1.0, 0.0]
class = 1
Renewed <= 1.807
gini = 0.109
samples = 65.0%
value = [0.942, 0.058]
class = 1
gini = 0.0
samples = 2.5%
value = [0.0, 1.0]
class = 3
Length <= -0.713
gini = 0.04
samples = 61.2%
value = [0.98, 0.02]
class = 1
Week <= -0.538
gini = 0.444
samples = 3.8%
value = [0.333, 0.667]
class = 3
AIDS <= -0.1
gini = 0.375
samples = 5.0%
value = [0.75, 0.25]
class = 1
gini = 0.0
samples = 56.2%
value = [1.0, 0.0]
class = 1
gini = 0.0
samples = 2.5%
value = [0.0, 1.0]
class = 3
gini = 0.0
samples = 1.2%
value = [1.0, 0.0]
class = 1
gini = 0.0
samples = 1.2%
value = [0.0, 1.0]
class = 3
gini = 0.0
samples = 3.8%
value = [1.0, 0.0]
class = 1



Спасибо за внимание!