

План

1. Установка необходимого ПО

1.1 Conda

1.2 Jupyter Notebook

1.3 PyCharm

Conda

Conda vs Pip

1. Pip - универсальный менеджер пакетов для Python и он не может обрабатывать библиотечные зависимости за пределами пакетов Python.
2. Pip собирает библиотеку из исходников. Часто приходится вручную устанавливать зависимые библиотеки.
3. Conda - это менеджер пакетов для любого программного обеспечения (не ограничивается только Python пакетами).
4. Conda устанавливает уже скомпилированную библиотеку со всем необходимыми зависимостями.

Miniconda vs Anaconda

1. Miniconda содержит conda (сам менеджер пакетов), Python и минимальный набор полезных пакетов.
2. Anaconda содержит все то же что и Miniconda, но помимо этого еще более 1500 пакетов, которые могут пригодиться. Anaconda может занимать намного больше места.

Установка Conda

1. Скачайте anaconda для своей ОС:
<https://www.anaconda.com/distribution/#download-section>
2. Если хотите установить miniconda, то ее можно скачать здесь:
<https://docs.conda.io/en/latest/miniconda.html>
3. Инструкции по установке для разных ОС:
<https://docs.anaconda.com/anaconda/install/>

Пример установки для Linux, macOS:

1. Скачиваете файл: "Anaconda3-2019.07-Linux-x86_64.sh" - конда для python3
2. В терминале:

```
>>> cd ~/Downloads/  
>>> bash Anaconda3-2019.07-Linux-x86_64.sh
```
3. Далее следуйте инструкциям по установке

Создаем виртуальное окружение

Пример создания окружения для первого ДЗ:

```
>>> conda create -n test_env python=3.6 requests  
>>> source activate test_env  
>>> source deactivate test_env
```

Пример установки numpy в выбранное окружение

```
>>> conda install -c anaconda numpy
```

In []:

Последовательность скриншотов с установкой конды

```
boris ~ > cd Downloads/  
boris ~ > Downloads > bash Anaconda3-2019.07-MacOSX-x86_64.sh  
  
Welcome to Anaconda3 2019.07  
  
In order to continue the installation process, please review the license  
agreement.  
Please, press ENTER to continue  
>>>  
  
Do you accept the license terms? [yes|no]  
[no] >>> yes  
  
Anaconda3 will now be installed into this location:  
/Users/boris/anaconda3  
  
- Press ENTER to confirm the location  
- Press CTRL-C to abort the installation  
- Or specify a different location below  
  
[/Users/boris/anaconda3] >>> _
```

```
done
installation finished.
Do you wish the installer to initialize Anaconda3
by running conda init? [yes|no]
[yes] >>> yes_
```

Проверка установки - доступные окружения и версия конды

```
boris ~ > Downloads conda env list
# conda environments:
#
base * /Users/boris/anaconda3

boris ~ > Downloads conda -V
conda 4.7.10

boris ~ > Downloads _
```

Установка окружения для первого задания

```
boris ~ > Downloads conda create -n test_env python=3.6 requests _
xz pkgs/main/osx-64::xz-5.2.4-h1de35cc_4
zlib pkgs/main/osx-64::zlib-1.2.11-h1de35cc_3

Proceed ([y]/n)? y_
```

```
boris ~ > Downloads conda env list
# conda environments:
#
base * /Users/boris/anaconda3
test_env /Users/boris/anaconda3/envs/test_env

boris ~ > Downloads _
```

Активация установленного окружения и запуск jupyter notebook

```
boris ~ source activate test_env
boris (e) test_env ~ conda env list
# conda environments:
#
base                /Users/boris/anaconda3
test_env             * /Users/boris/anaconda3/envs/test_env

boris (e) test_env ~ jupyter notebook
```

```
boris (e) test_env ~ which python
/Users/boris/anaconda3/envs/test_env/bin/python
boris (e) test_env ~ which jupyter
/Users/boris/anaconda3/bin/jupyter
```

```
boris (e) test_env ~ conda install jupyter
boris (e) test_env ~ which jupyter
/Users/boris/anaconda3/envs/test_env/bin/jupyter
```

In []:

In []:

In []:

In []:

Jupyter Notebook

Зачем нужен Jupyter Notebook?

Для запуска ввести следующую команду в командной строке:

```
>>> jupyter notebook
```

Два режима работы с ячейками:

1. Режим редактирования содержимого ячейки - Enter
2. Командный режим - Esc (При редактировании ячейки)

In []:

```
import numpy as np
```

```
In [ ]: # Посмотреть документацию к функции
        ?np.mean
```

```
In [ ]: # Посмотреть исходный код функции
        ??np.mean
```

```
In [ ]: # Shift+Tab - справка во время ввода аргументов
        np.mean()
```

```
In [ ]: # Shift+Tabx2 - документация к функции при вводе аргументов
        np.mean()
```

```
In [ ]: # Shift+Tabx3 - тоже самое что и предыдущий пункт,
        # только документация не закрывается в течении 10 секунд пока вы печатаете
        np.mean()
```

```
In [ ]: # Shift+Tabx4 - тоже самое что ?np.mean - полная документация
        np.mean()
```

Измерение времени выполнения ячейки

Обозначения

1. s - секунды
2. ms - мили == $10e-3$ s
3. μ s - микро == $10e-6$ s
4. ns - нано == $10e-9$ s

%%time - после выполнения ячейки выводит время за которое она отработала

```
In [ ]: def func(n: int = 10**6):
        print('Function is called')
        return [i**2 for i in range(n)]
```

```
In [ ]: %%time
        _ = func(10**7)
```

%%timeit - делает несколько запусков ячейки и выводит mean \pm std

```
In [ ]: %%timeit
        _ = func()
```

```
In [ ]: %%timeit -n 1 -r 1
        _ = func(10**6)
```

```
In [ ]: %time _ = func(10**6)
        %time _ = func(10**5)
        %timeit _ = func(10**6)
```

```
In [ ]:
```

Выполнить команду командной строки

Клонируем репозиторий для выполнения первого задания

В нем есть файл, который оценивает ваш классификатор, а также пример классификатора.

```
In [ ]: ! git clone https://github.com/nvanva/filimdb_evaluation.git
```

Распакуем файлы с текстами

```
In [ ]: ! tar xvfz filimdb_evaluation/FILIMDB.tar.gz -C filimdb_evaluation/
```

```
In [ ]: ! ls filimdb_evaluation/FILIMDB/
```

TEX

В решениях домашних заданий придется писать формулы, для этого можно пользоваться tex-ом, например <https://www.overleaf.com>. Но иногда может понадобиться запустить кусок кода, и тогда удобно использовать jupyter как редактор tex-a и одновременно интерпретатор питона.

$$c = \sqrt{a^2 + b^2}$$

Печать в PDF

Либо File > download as > .pdf, но работает не всегда.

Либо в хrome, и в других браузерах есть печать страницы - ctrl+P. В хrome работает хорошо.

Статья об особенностях jupyter notebook:

<https://habr.com/ru/company/wunderfund/blog/316826/>

```
In [ ]:
```

```
In [ ]:
```

PyCharm

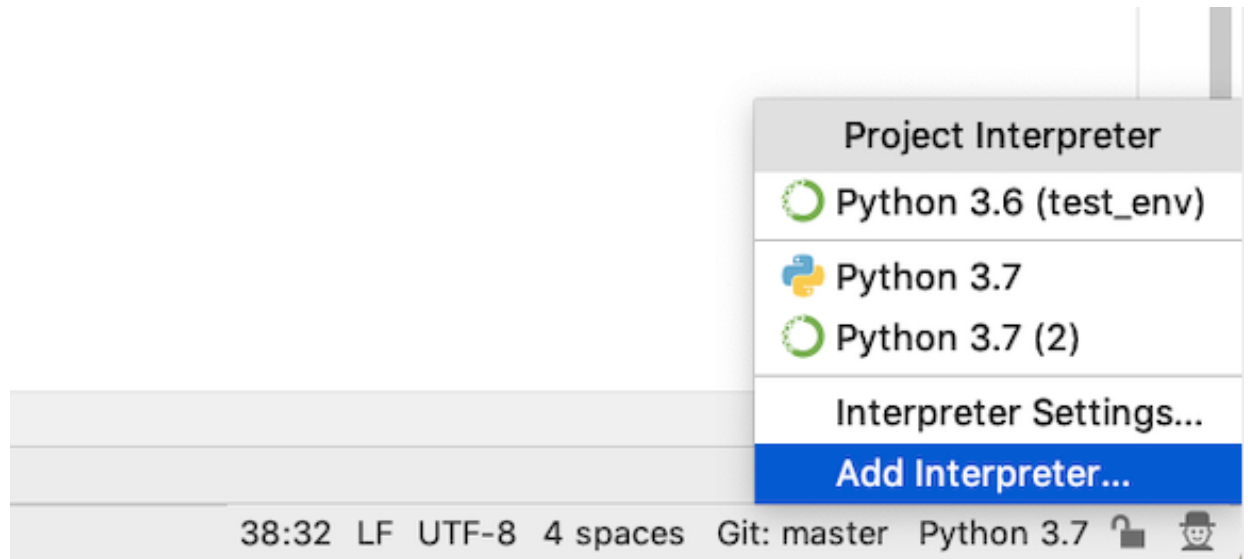
PyCharm - среда разработки для питона, есть отладчик и анализ кода.

Ссылка для скачивания под Вашу ОС: <https://www.jetbrains.com/pycharm/download/>

1. Затем откроем скачанный проект в PyCharm
2. И настроим его так, чтобы вызывался интерпретатор питона из созданного ранее окружения

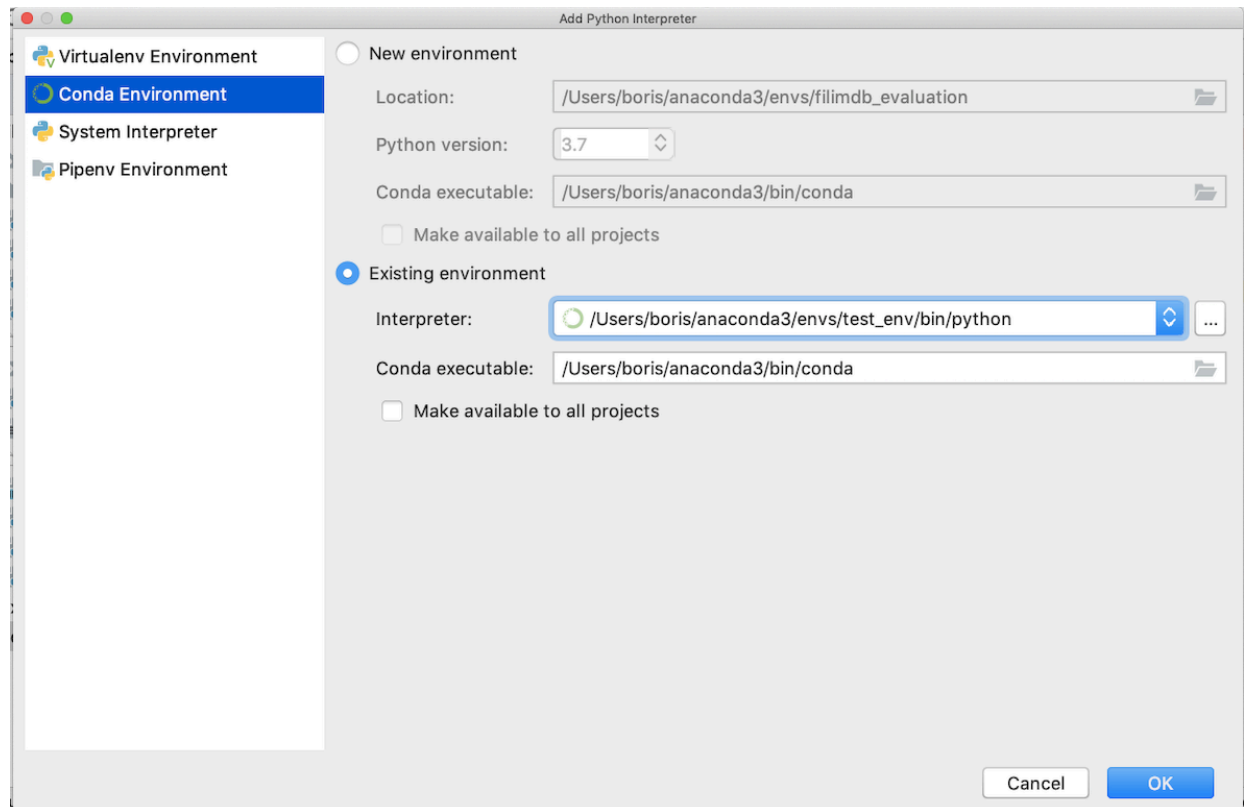
Чтобы открыть настроить PyCharm под виртуальное окружение анаконды, нужно:

1. В правом нижнем углу найти Python
2. Нажать на него, а затем нажать Add Interpreter

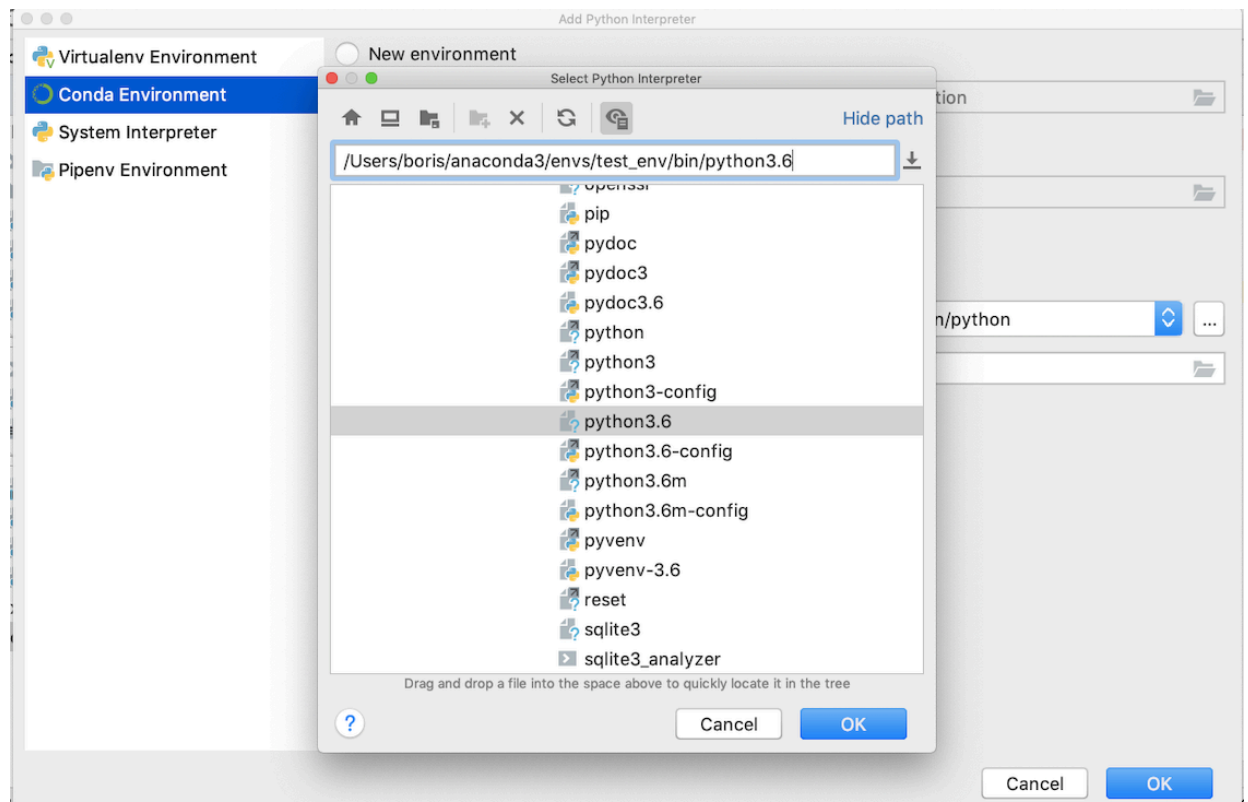


3. Выбрать conda environment

4. Далее либо выбрать existing environment, либо создать новое



5. Ввести путь до интерпретатора питона в нужном окружении (По аналогии с тем что на картинке)

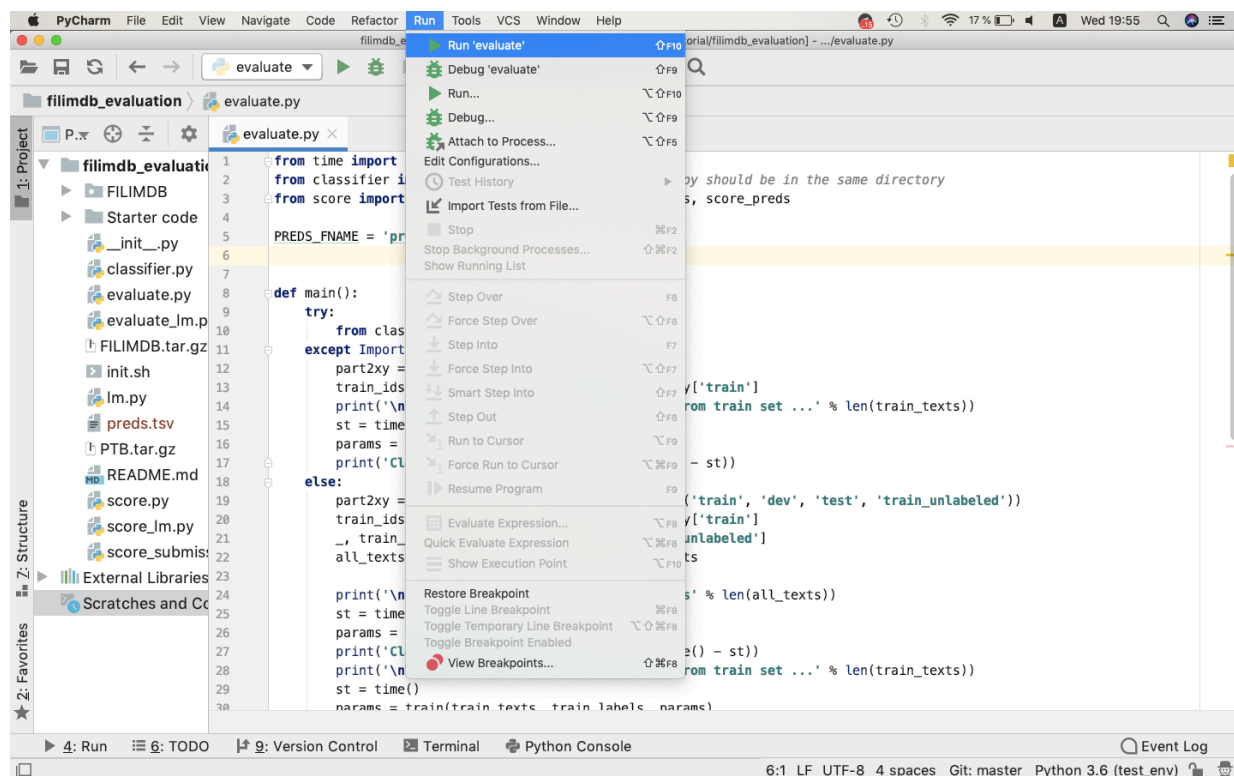


In []:

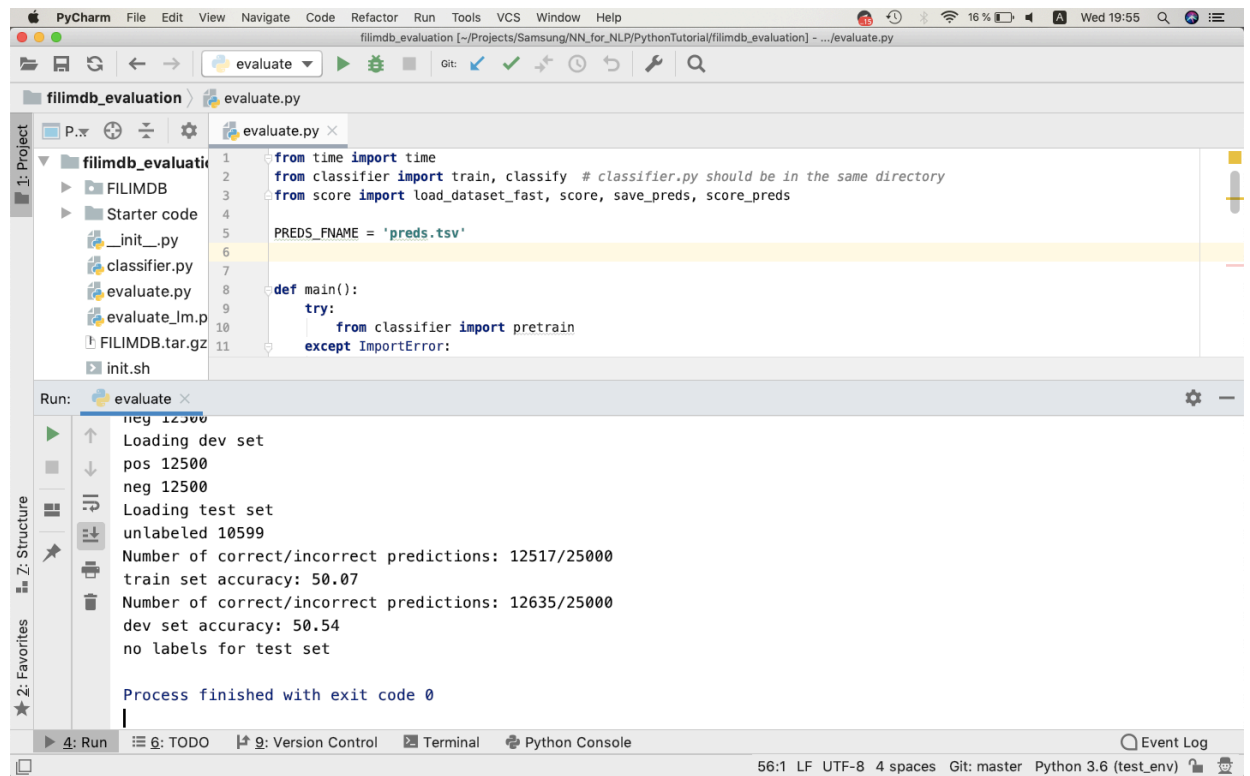
Оценка работы классификатора для первого задания

После того как открыли загруженный проект в PyCharm и настроили интерпретатор можно запустить классификатор из репозитория

Запускаем файл `evaluate.py`, он сам импортирует нужные функции из классификатора



После этого видим в окне снизу результаты работы нашего классификатора



The screenshot shows the PyCharm IDE interface. The main editor displays the `evaluate.py` file with the following code:

```
1 from time import time
2 from classifier import train, classify # classifier.py should be in the same directory
3 from score import load_dataset_fast, score, save_preds, score_preds
4
5 PRED_S_FNAME = 'preds.tsv'
6
7
8 def main():
9     try:
10         from classifier import pretrain
11     except ImportError:
```

The Run window at the bottom shows the output of the `evaluate` script:

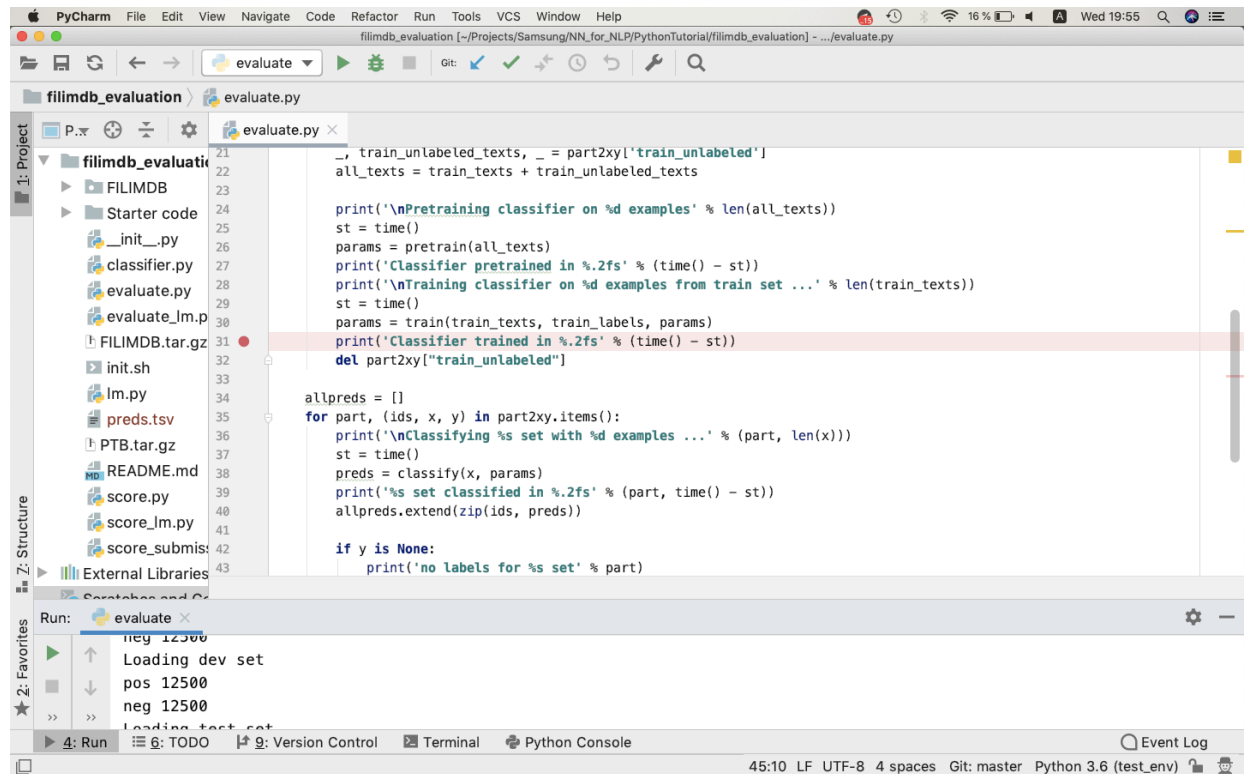
```
neg 12500
Loading dev set
pos 12500
neg 12500
Loading test set
unlabeled 10599
Number of correct/incorrect predictions: 12517/25000
train set accuracy: 50.07
Number of correct/incorrect predictions: 12635/25000
dev set accuracy: 50.54
no labels for test set

Process finished with exit code 0
```

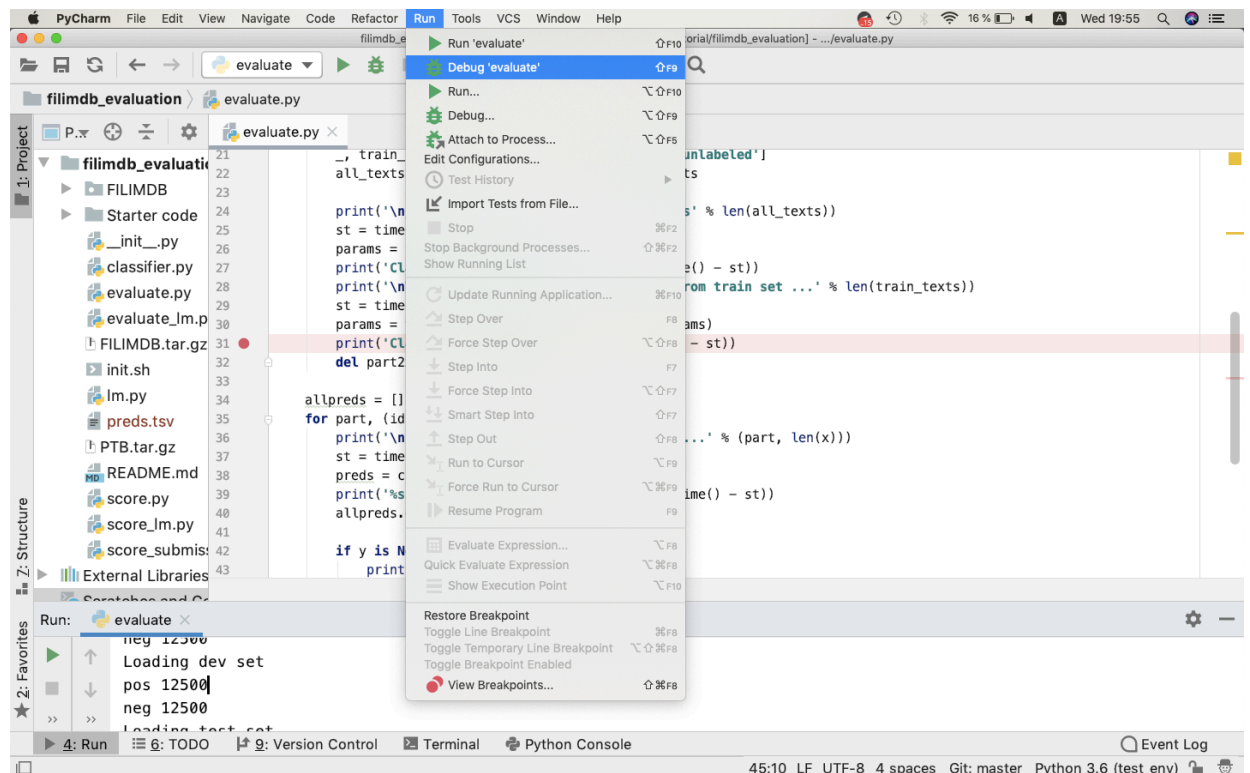
In []:

Пример использования отладчика в PyCharm

1. Ставим метку как на картинке, нажав чуть правее от номера строки - 31



2. Запускаем отладку нажав debug



3. В окне ниже можем видеть значения переменных в момент когда выполнение программы дошло до поставленной метки

