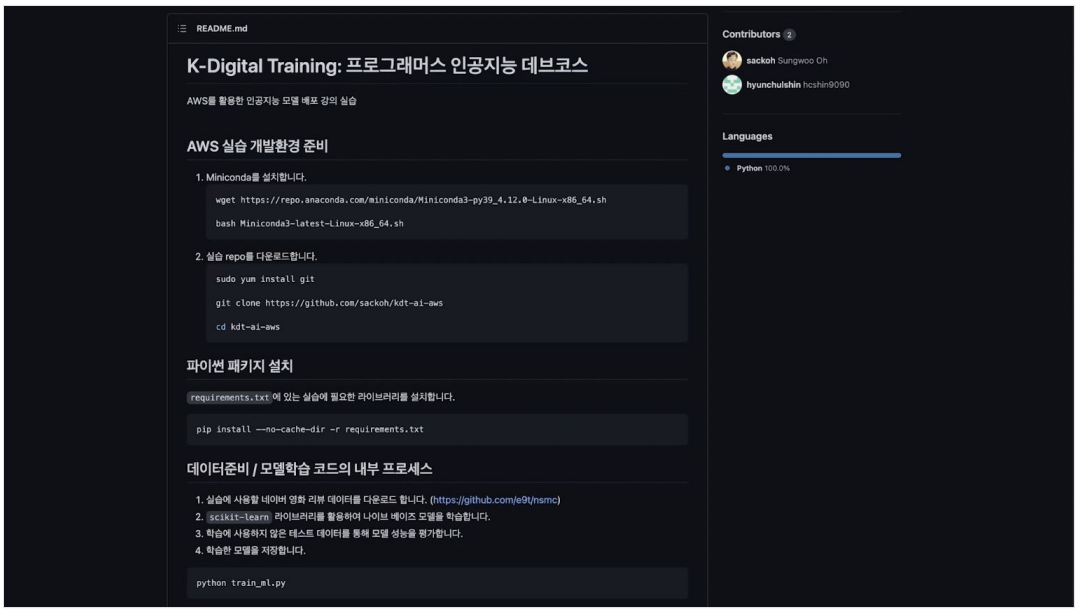
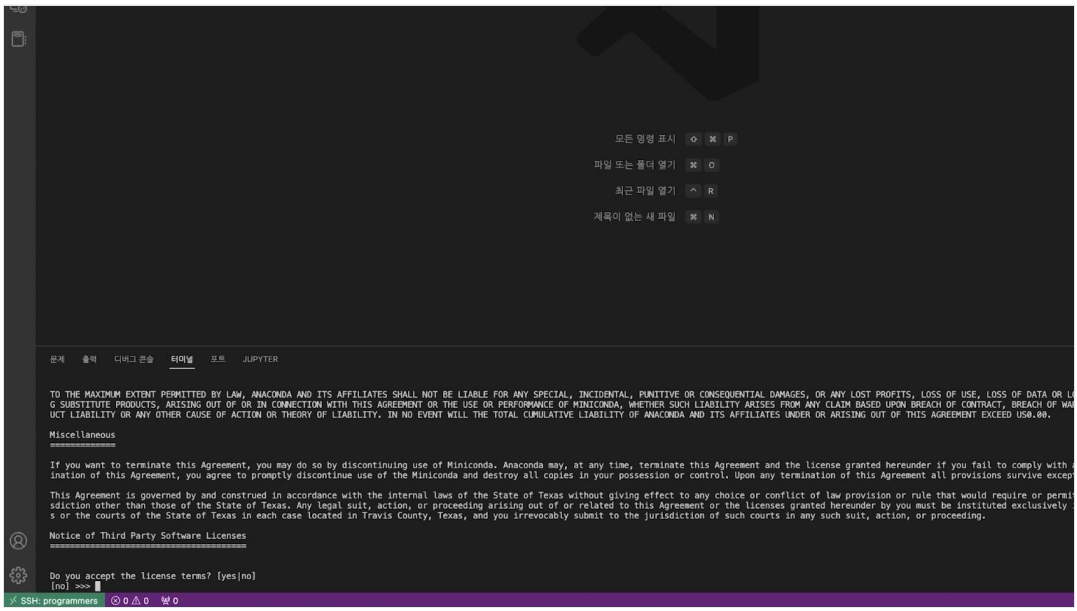
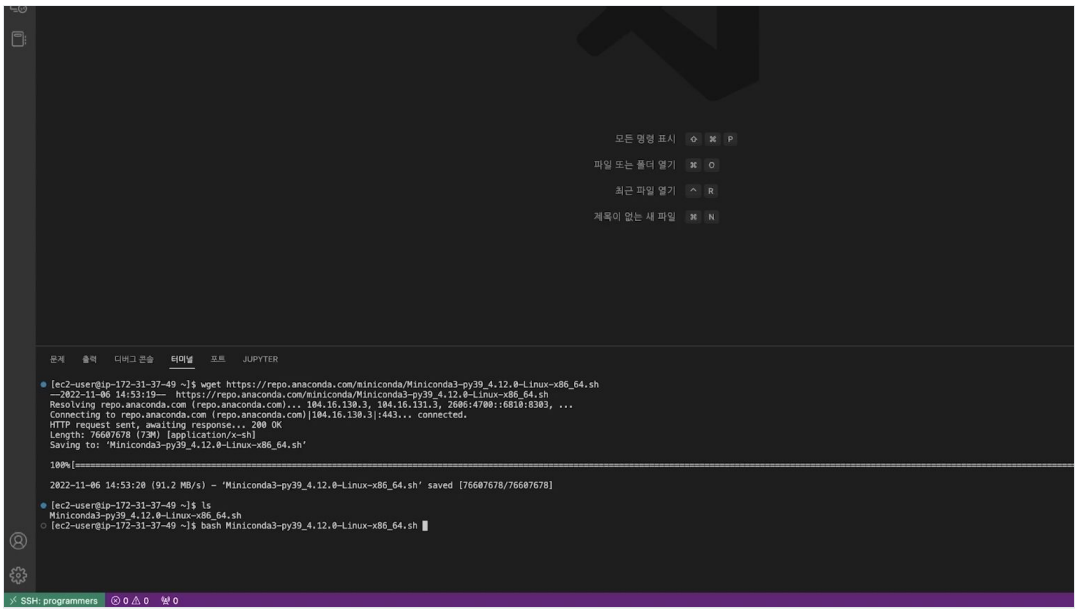
https://github.com/sackoh/kdt-ai-aws



우리가 만들었던 아마존 인스턴스 환경에서 미니콘다를 다운받고(1번째 코드) 설치한다.(2번째 코드)



q를 눌러주면 라이센스에 동의하는지 화면으로 바로 내려간다.(약관 바로 아랫줄로 간다)

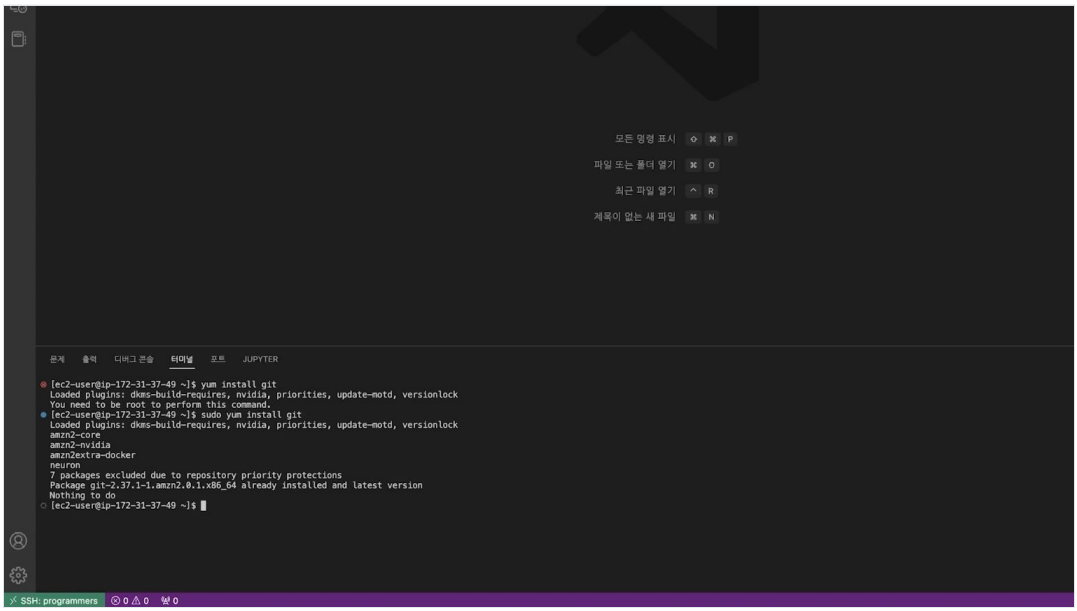


yes를 타이핑 그리고 어디에 로케이션을 잡을지 인대 enter누르면 된다 그럼 설치가 진행된다

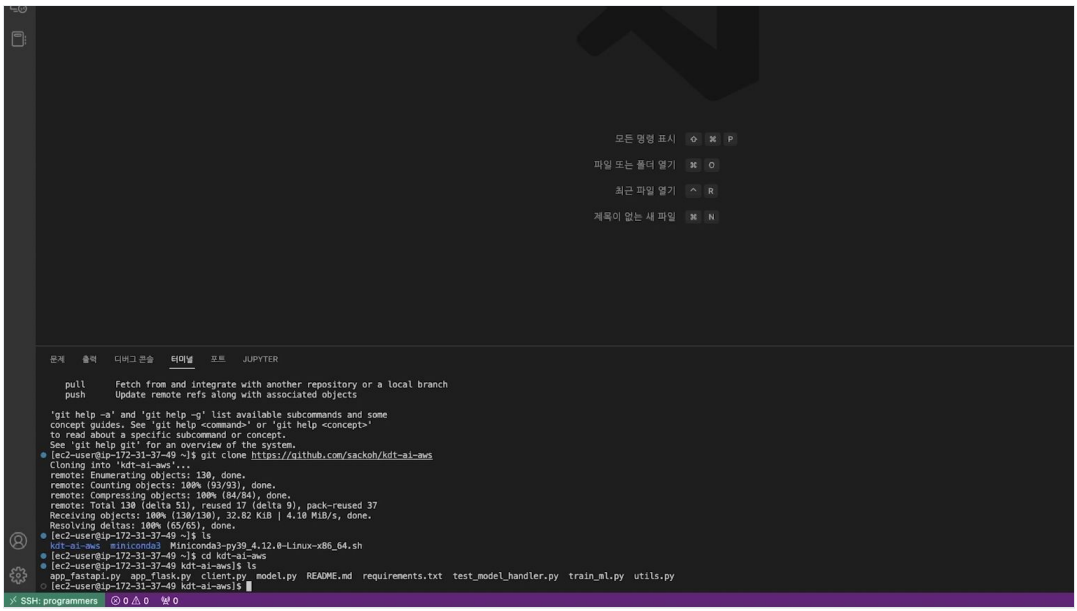
또 yes누르고 그럼 설치가 끝남

미니콘다는 아나콘다에서 필수적인 기능들만 빼서 사이즈를 많이줄인 패키지 이름이다

인스턴스에 git을 먼저 설치하자



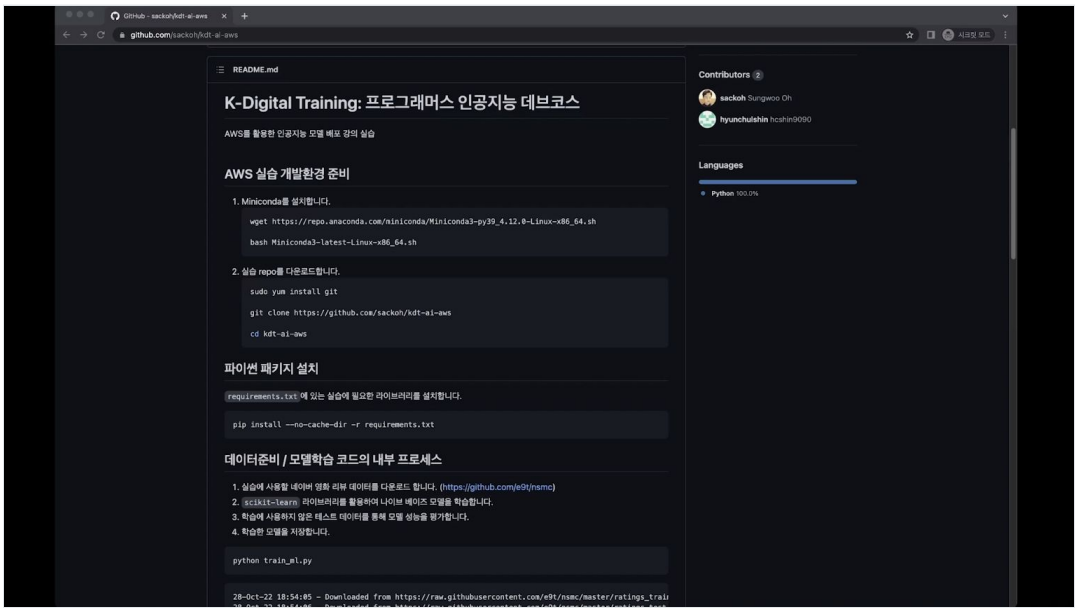
이제 강의 관련된 실습코드를 다운로드 받도록 한다.



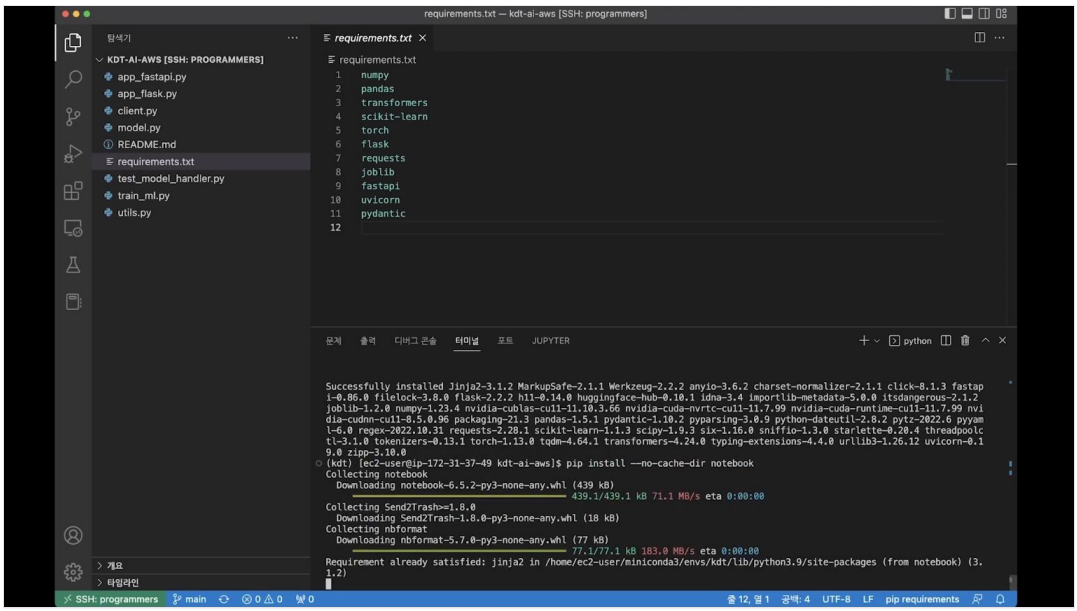
먼저 가상환경을 만든다 가상환경 이름은 kdt, 파이썬 버젼은 3.9를 사용

그리고 pip install --no-cache-dir -r requirements.txt 명령어를 사용해서 인스턴스의 공간이 녹록치 않기 때문에 파이썬 설치 파일들을 캐시로 저장하지 않도록 하는 인자를 추가해서 필요한 라이브러리 설치를 진행한다

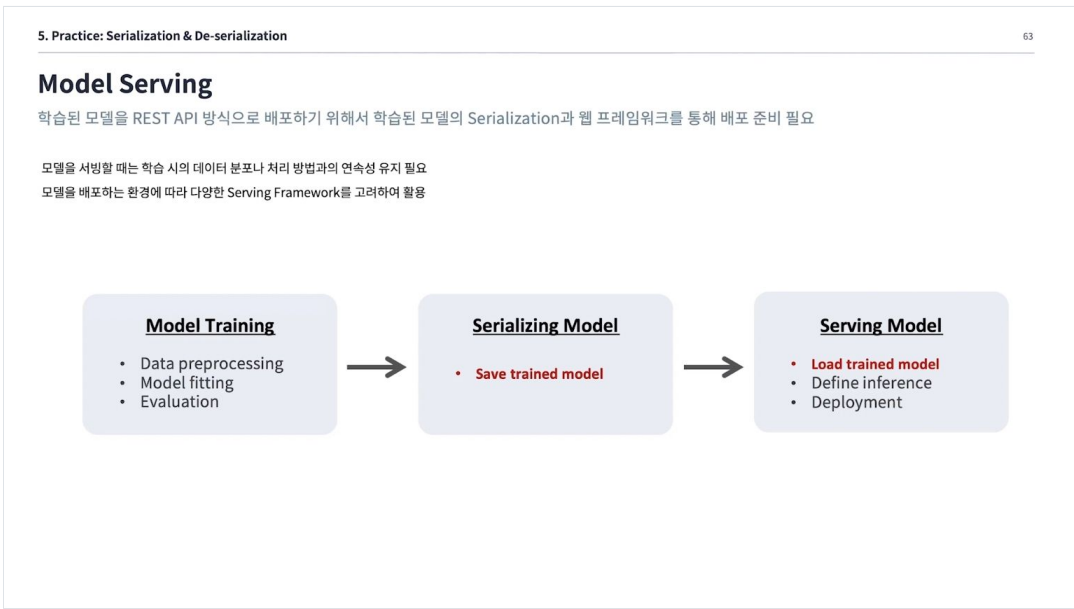
aws를 사용하는 장점중 하나가 global infrastructure를 통한 고속 네트워킹이 가능하다는 건대 실제 설치 속도가 초당 100메가가 나온다고 함



쥬피터 노트북도 설치 진행



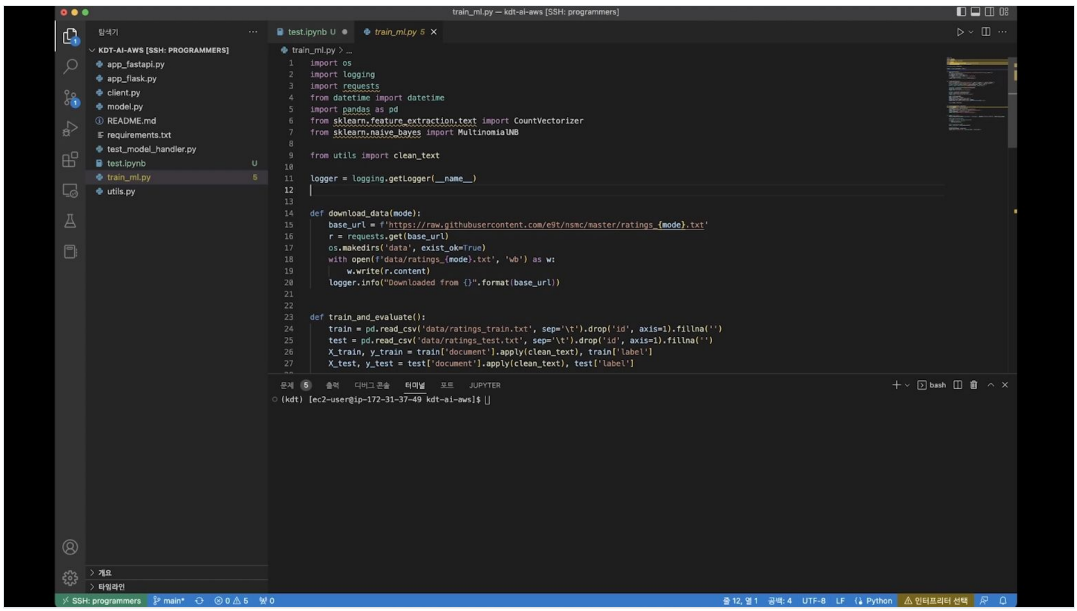
test.ipynb 노트북 파일을 만든다 -> 커널은 우리가 만들었던 가상환경 kdt를 선택!



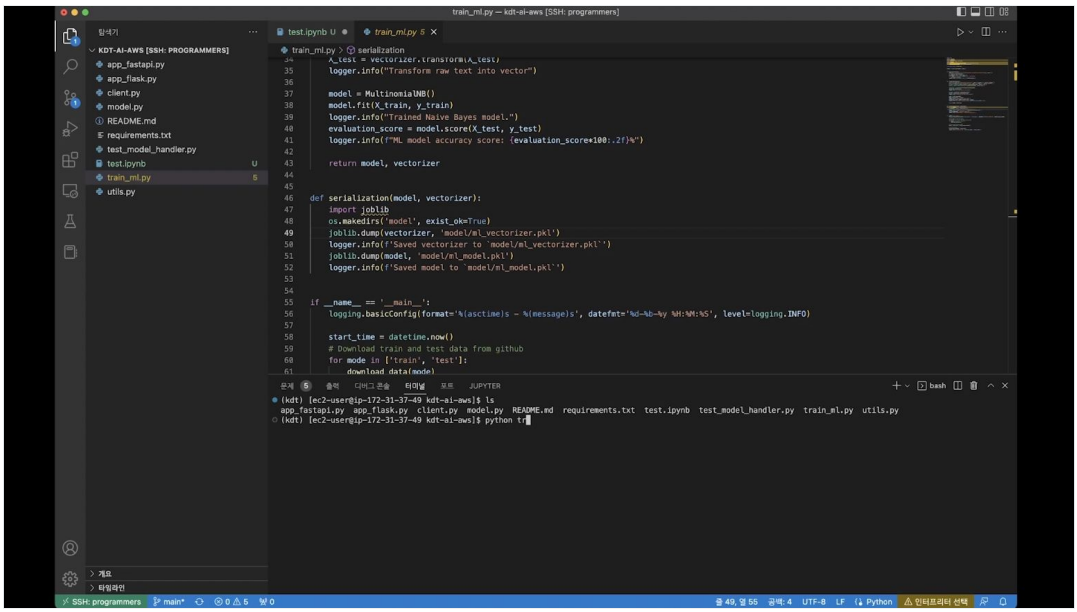
머신러닝 모델 학습

실습 진행을 위해 사전 준비한 코드를 실행하여 모델 학습 및 저장

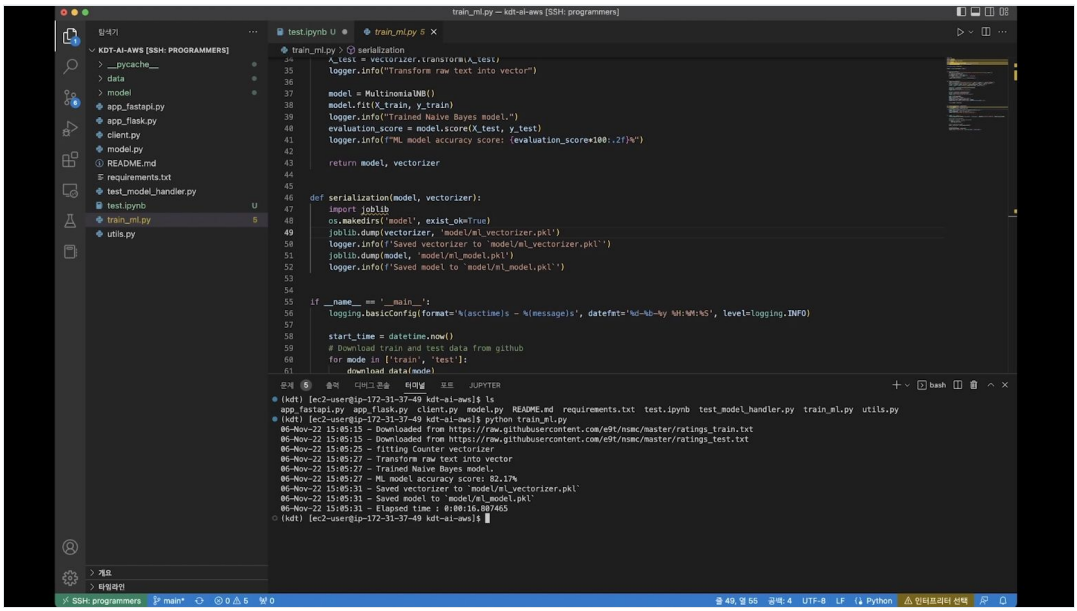
```
$ python train_ml.py
```



소스코드는 나이브 베이지안을 이용해서 분류기를 만드는 방식, 우리가 실습으로 다루고자 하는 부분은 def serializaton()이다. jolib이라는 패키지를 이용해서 모델을 직렬화 한다. 따라서 학습하는 모델에서 모델 직렬화와 저장까지 한번에 이루어 지도록한다.



학습하는 모델은 네이버 감성분류 데이터이다



모델 2개가 저장되었다. 벡터라이저와 모델 2개가 저장되었다. vetorizer의 역할은 string인 텍스트를 벡터로 변환하는 변환기, 실질적으로 모델을 분류하는 모델파일은 ml_model.pkl이다. 데이터 다운 부터 학습, 모델 생성까지 16초정도 소요되었다.

모델을 돌리면 이렇게 다운받은 txt파일 2개와 모델파일 2개가 생성된 것을 확인할 수있다. 직렬화된 파일은 보통 바이트 형식으로 변환이 되기 때문에 일반적인 에디터에서 보기 어렵다.

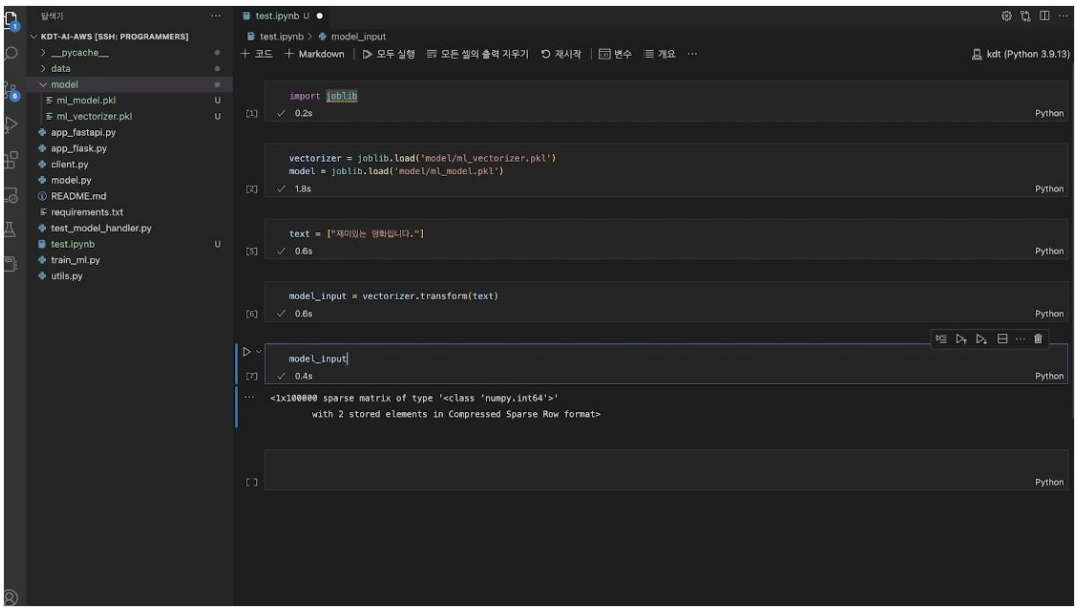
이제는 학습한 모델을 어떻게 불러오는지 살펴볼 것이다

De-serialization

저장된 모델을 불러와 특정 입력 값에 대한 예측 수행

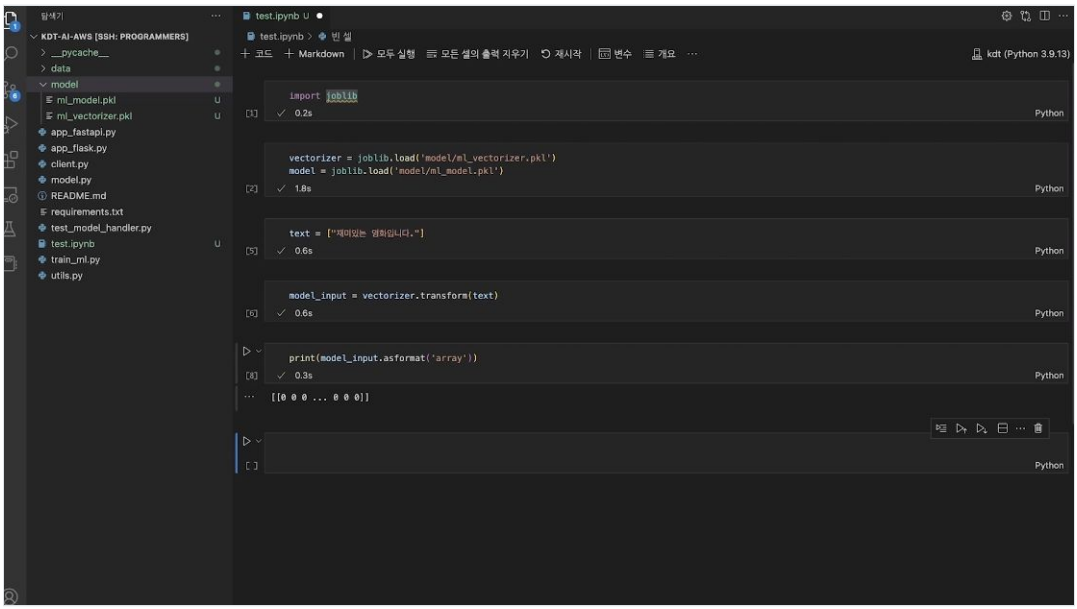
1. 터미널 환경에서 python 또는 ipython, jupyter notebook 실행
2. 아래 예제 코드 테스트하여 de-serialization 확인

```python
# De-serialization
import joblib
vectorizer = joblib.load('model/ml_vectorizer.pkl')
model = joblib.load('model/ml_model.pkl')

# Sample text
text = '재미있는 영화입니다.'

# Test loaded model and vectorizer
model_input = vectorizer.transform(text)
print(model_input.asformat('array'))

model_output = model.predict_proba(model_input)
model_output = model_output.argmax(axis=1)
id2label = {0: 'negative', 1: 'positive'}
print(f'sentiment : {id2label[model_output[0]]}')
```

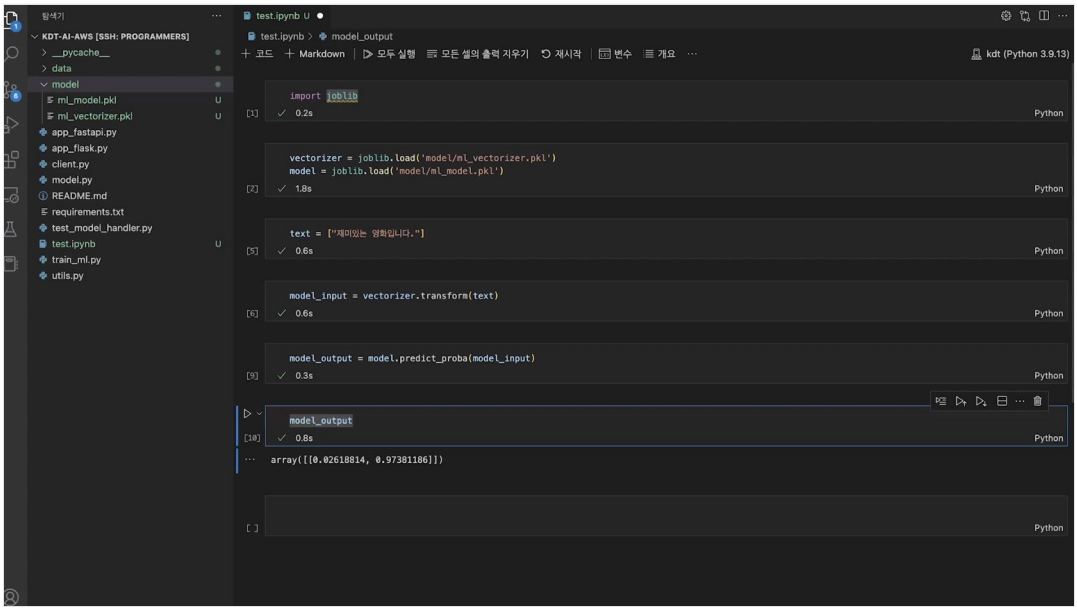


sparse matrix는 희소행렬이라고도 하는대 여기서 사용하고 있는 벡터는 특정 키워드들이 사전으로 존재하고 이 모델같은 경우는 10만개가 될 것이다 이 10만개의 feature중에서 해당 사전의 단어가 있는지 없는지 0과1로 구분된 martrix가 있는거고 그러기 때문에 0이 굉장히 많기 때문에 우리가 0은 연산을 할 때 필요가 없으므로 필요없는 0의 연산을 구분하기 위해서 sparse matrix자료구조를 사용하게 된다.

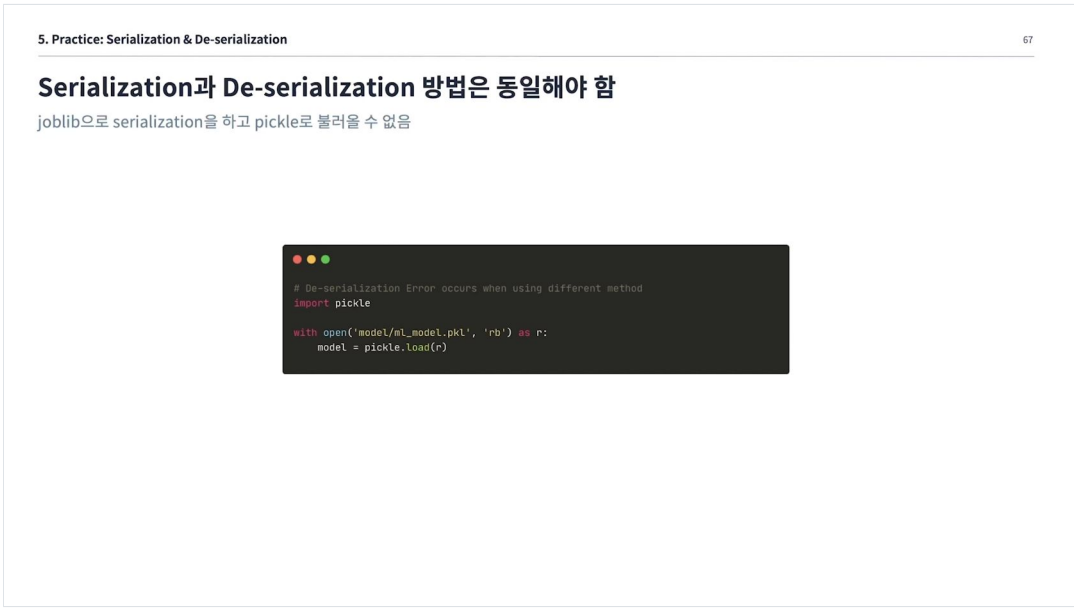
vectorized된 input을 살펴보자! sparse matrix때문에 array로 변환해서 살펴보아야 한다!

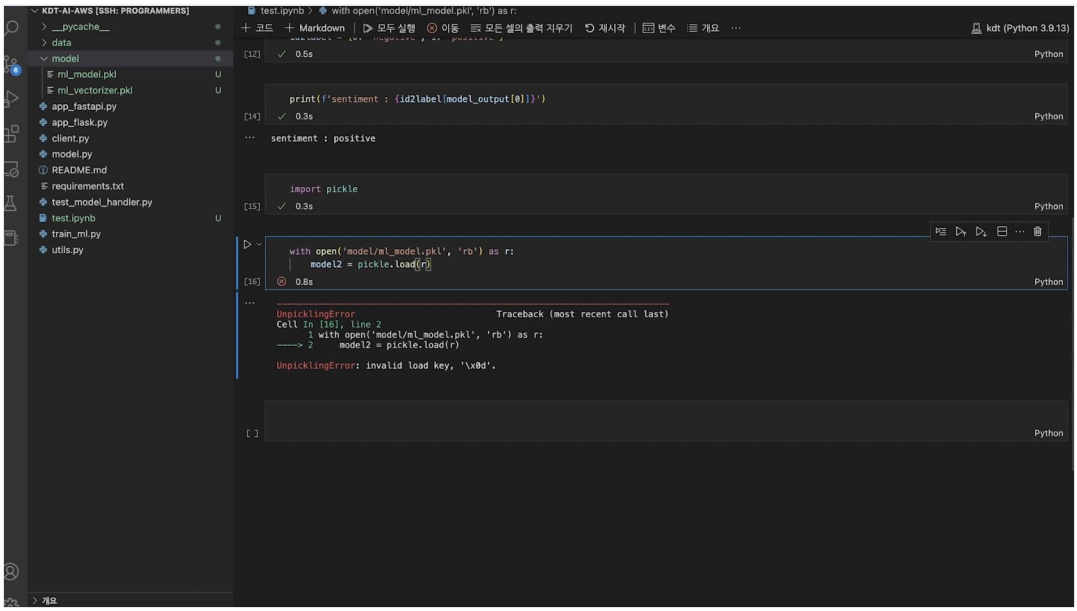


살펴본 결과 한 눈에 살펴보기에는 어렵다(0만 보이므로)



모델의 예측결과를 확인해보면 확률분포 형태로 나오고 결과를 보면 위치에 따라서 label이 정해지는대 2번째 값의 확률이 압도적으로 높으므로 긍정(positive)이 나오게 된다. "재미있는 영화입니다"에 대해서 긍정으로 잘 예측한 모습이다!





보면 unpicking error가 발생했다. joblib으로 저장했기 때문에 serialization방식이 조금 달라서 pickle의 load메서드로 가져오려고 할때 에러가 발생하는 것이다.

따라서 실제 현업에서 모델은 내가 개발하고 서빙은 다른사람이 한다고 했을 때 어떤 프레임워크나 라이브러리를 사용해서 serialization을 했는지 API명세를 할 때 모델의 대한 정보도 같이 제공해야 한다!