

Part 8. FastAPI 기반 감성분석 API개발



FastAPI로 개발하는 방식을 설명할 것이다. 최근에 많이 떠오르는 framework로 flask를 대체해서 머신러닝 분야에서 모델을 serving할 때 사용하고 있다. 빠르게 개발할 수 있다는 점과 좋은 성능을 보인다는 장점이 있다. 실습에 사용하는 데이터는 네이버 영화리뷰이다.

8. FastAPI 기반 감성분석 API 개발

97

네이버 영화리뷰 감성분석 개요

네이버 영화리뷰 데이터로 학습한 ML/DL 모델을 활용해 감성분석 API 개발

나이트비제이안 모델과 딥러닝 모델로 학습한 두 개의 모델을 서빙하며 0은 부정성 1은 긍정성을 의미
학습에 사용한 데이터는 박은정님이 공개한 NSMC 데이터

id	document	label
9976970	아 더빙.. 진짜 짜증나네요 목소리	0
3819312	흠...포스터보고 초딩영화줄....오버연기조차 가볍지 않구나	1
10265843	너무재밌었다그래서보는것을추천한다	0
9045019	교도소 이야기구먼 ..솔직히 재미는 없다..평점 조정	0
6483659	사이몬페그의 익살스런 연기가 돋보였던 영화!스파이더맨에서 늙어보이기만 했던 커스틴 ...	1
5403919	막 걸음마 떤 3세부터 초등학교 1학년생인 8살용영화. ㅋㅋㅋ...별반개도 아까움.	0
7797314	원작의 긴장감을 제대로 살려내지못했다.	0

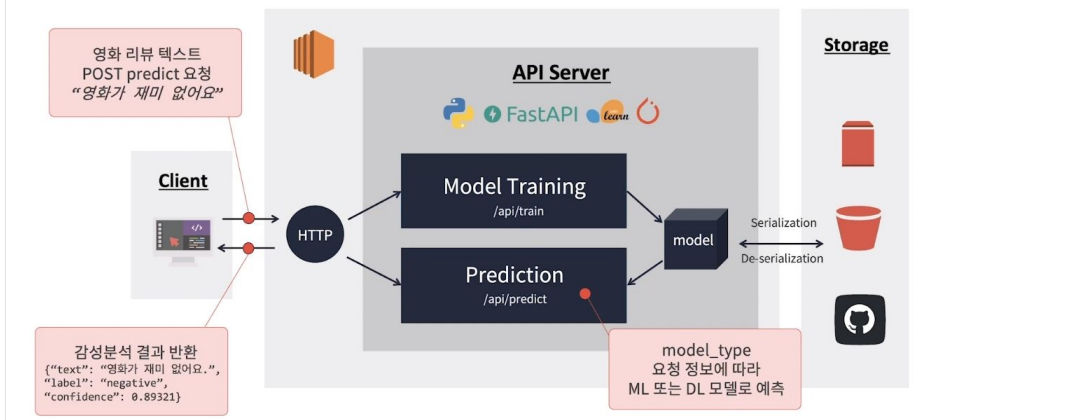
Source: <https://github.com/ekik/nsmc/>

api개발 방식도 동일하지만 웹 프레임워크로 FastAPI를 사용할 것이다.

감성분석 API 개발 방향

AWS EC2와 Python FastAPI 기반 모델 학습 및 추론을 요청/응답하는 API 서버 개발

Copied



성능이나 안정성을 인정받은 웹 프레임워크이다.

NodeJS나 Go와 같이 API배포나 서버배포를 위해 사용되는 언어와 비교했을 때에도 파이썬에서도 뒤지지 않는 성능을 낼 수 있도록 알려짐

FastAPI

FastAPI는 Python 웹 개발 환경에서 Flask와 같이 간단하게 애플리케이션을 만들 수 있는 micro framework
Netflix, Uber, Microsoft 같은 글로벌 기업 뿐만 아니라 국내 많은 기업들도 production 용으로 FastAPI를 사용하고 있음

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.7+ based on standard Python type hints.

잘 정리된 Documentation을 통해 production 수준의 코드를 개발하고 배포하는데 무리가 없음

Data Science 및 Machine Learning 분야에서 API 서비스를 개발하고 제공할 때에도 많이 사용되고 있음

NodeJS나 Go와 같은 언어와 비교했을 때에도 성능이 뒤처지지 않는 것으로 알려짐



The key features are:

- **Fast:** Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic). One of the fastest Python frameworks available.
- **Fast to code:** Increase the speed to develop features by about 200% to 300%.*
- **Fewer bugs:** Reduce about 40% of human (developer) induced errors.*
- **Intuitive:** Great editor support. Completion everywhere. Less time debugging.
- **Easy:** Designed to be easy to use and learn. Less time reading docs.
- **Short:** Minimize code duplication. Multiple features from each parameter declaration. Fewer bugs.
- **Robust:** Get production-ready code. With automatic interactive documentation.
- **Standards-based:** Based on (and fully compatible with) the open standards for APIs: **OpenAPI** [1-4] (previously known as **Swagger**) and **JSON Schema** [5-7].

* estimation based on tests on an internal development team, building production applications.

Source: <https://fastapi.tiangolo.com>

FastAPI가 Flask보다 빠른 성능을 보일 수 있었던 이유중 하나는 Flask는 WSGI방식으로 통신을 한다. 이전 실습에서도 API서버를 돌릴때 빨간색으로 나온게 WSGI이다

WSGI는 한번에 하나의 응답만 처리를 하도록 고안이 되어 있어서 요청이 왔을때 결과값을 바로 리턴해주는것을 전제로 하고있다. 이러한 동기적인 프로세스를 전제로 하는 인터페이스 표준이다 보니 성능이나 효율성이 떨어지게 된다.

WSGI의 한계를 극복하기 위해 나온것이 ASGI이고 ASGI는 비동기적인 처리를 가능하게 된 인터페이스 표준이다. 서로 비슷해서 WSGI코드를 ASGI코드로 수정할때 많은 코드수정이 발생하지 않는다.

파이썬에서 코루팅(?)을 지원한다 원래 yield하는 방식으로 했다면 이젠 async와 await로 적용할 수있도록 파이썬 명령어가 변경되었다.

WSGI에서 ASGI로 변경하는것은 async와 await키워드를 이용하면 비동기적으로 호출하고 그 결과를 먼저나오는 결과를 반환하는 형태로 동작하게 된다.

WSGI → ASGI

WSGI(Web Server Gateway Interface)는 Python에서 애플리케이션과 서버가 서로 통신하기 위해 2003년 처음 도입된 인터페이스 표준

ASGI(Asynchronous Server Gateway Interface)는 WSGI처럼 공통적인 인터페이스가 기술되어 있지만 애플리케이션당 여러 개의 비동기 이벤트를 허용한다는 차이가 있음

WSGI는 한 번에 하나의 요청과 응답만 처리하도록 고안되었으며, 요청이 왔을 때 응답이 즉시 return되는 것을 전제로 하고 있음
이러한 동기 동작을 전제하는 인터페이스 표준은 비동기 작업을 처리하기에는 성능이나 효율성이 떨어지며,
동시에 웹소켓과 같은 고급 프로토콜을 효과적으로 처리하기 어렵다는 것을 의미함

ASGI는 표면적으로는 WSGI와 비슷해서 WSGI 기반의 웹 프레임워크에서 개발한 소스코드를 쉽게 migration 가능함
async 함수와 비동기 처리에 친화적인 웹 프레임워크로는 Starlette, FastAPI가 대표적이고 Django 3.0 이상에서도 ASGI를 지원함



documentation을 생성하는 기능과 요청된 데이터에 대해서 validation하는 기능이 있어서 머신러닝과 같이 입력 데이터가 항상 발생하고 그 입력데이터가 그 입력 데이터가 api에서 정의한 형태가 맞게 실제로 요청이 되는지 확인하는 코드를 별도의 코드를 개발하지 않고도 적용 할 수있다.

Versus Flask

FastAPI는 Flask와 비교했을 때 ASGI를 지원하여 더 빠른 성능을 보여준다는 것 외에도 다양한 장점이 있음

Built-in으로 제공되는 Documentation generation 기능과 Data validation 기능은 Machine Learning API 개발 측면에서 유용함

반복되는 Error Analysis와 Qualitative model inference testing을 위해서 유용한 기능들을 built-in으로 제공하고 있으며 이는 개발자가 실수할 수 있는 확률을 줄여줌

- Built-in Document Generation: Swagger UI를 기반으로 개발한 API를 Browser를 통해 테스트하고 documentation을 확인할 수 있음
- Built-in Data Validation: API endpoint를 통해 입력된 데이터의 type을 Pydantic 라이브러리를 활용해 체크할 수 있도록 함

Source: <https://fastapi.tiangolo.com>

document는 broser에서 swagger ui 형태로 나오고 data validation은 Pydantic 라이브러리를 활용해서 데이터 타입을 확인하게 된다

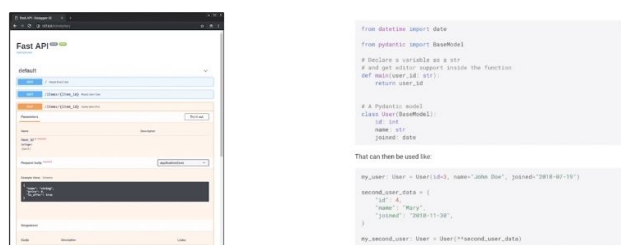
Versus Flask

FastAPI는 Flask와 비교했을 때 ASGI를 지원하여 더 빠른 성능을 보여준다는 것 외에도 다양한 장점이 있음

Built-in으로 제공되는 Documentation generation 기능과 Data validation 기능은 Machine Learning API 개발 측면에서 유용함

반복되는 Error Analysis와 Qualitative model inference testing을 위해서 유용한 기능들을 built-in으로 제공하고 있으며 이는 개발자가 실수할 수 있는 확률을 줄여줌

- Built-in Document Generation: Swagger UI를 기반으로 개발한 API를 Browser를 통해 테스트하고 documentation을 확인할 수 있음
- Built-in Data Validation: API endpoint를 통해 입력된 데이터의 type을 Pydantic 라이브러리를 활용해 체크할 수 있도록 함



Source: <https://fastapi.tiangolo.com>

FastAPI 기반 API 개발 & 배포

Model을 전역변수로 불러오고 요청된 텍스트에 대해 예측 결과를 반환하는 코드 입력

```
from typing import List, Dict, Union
from fastapi import FastAPI
from pydantic import BaseModel
from uvicorn import Config, Server
import os

app = FastAPI()

# model_loader as global variable [2 LINES]
...

# define request/response data type for validation
class Request(BaseModel):
    text: str = Field(..., description="Text to predict")
    model_type: str

class Response(BaseModel):
    # prediction: List[str] = Field(..., description="Predictions", min_items=1, max_items=10)
    prediction: List[str]

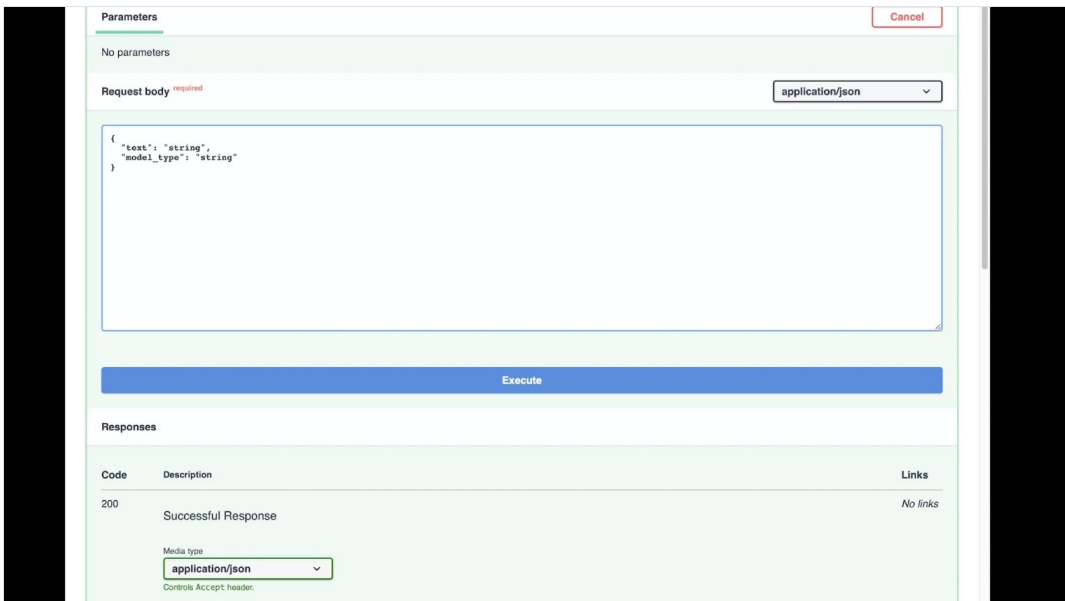
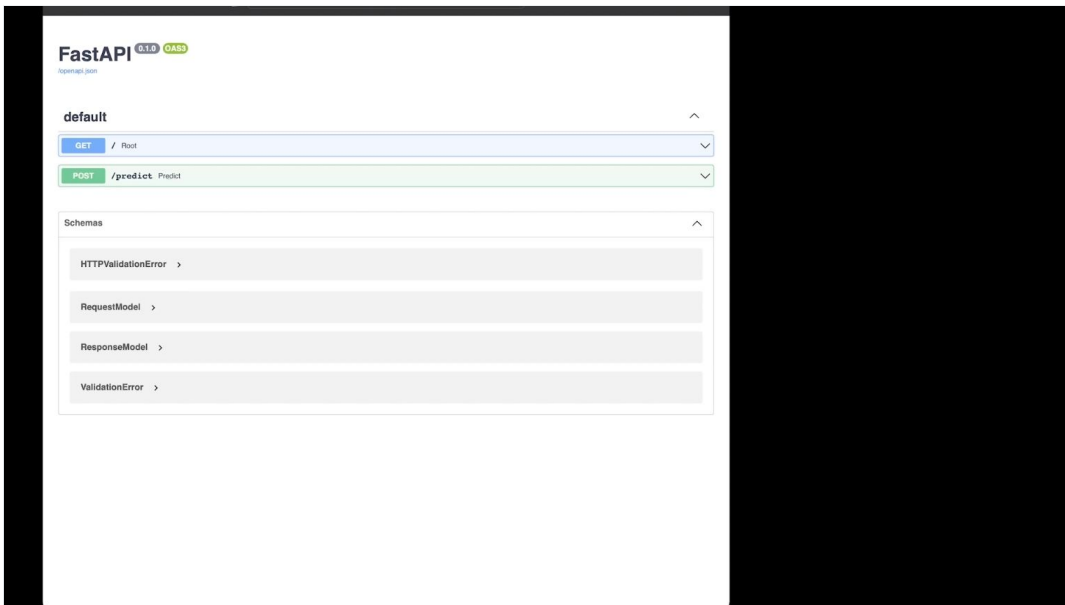
@app.get("/")
async def root():
    return {"message": "Hello World"}

@app.post("/predict", response_model=Response)
async def predict(request: Request):
    text = request.text
    model_type = request.model_type

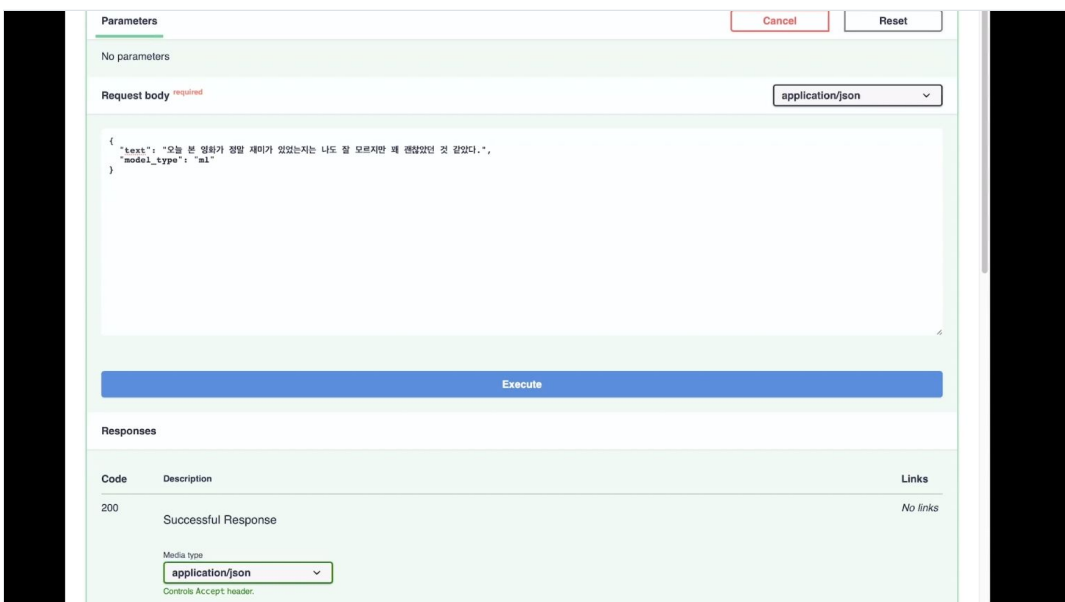
    # model_loader [2 LINES]
    if model_type == 'ml':
        ...
    else:
        ...

    # response
    result = {'text': text, 'label': 1, 'confidence': 0.9}
    # For 1, 0, -1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839, 1840, 1841, 1842, 1843, 1844, 1845, 1846, 1847, 1848, 1849, 1850, 1851, 1852, 1853, 1854, 1855, 1856, 1857, 1858, 1859, 1860, 1861, 1862, 1863, 1864, 1865, 1866, 1867, 1868, 1869, 1870, 1871, 1872, 1873, 1874, 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 1884, 1885, 1886, 1887, 1888, 1889, 1890, 1891, 1892, 1893, 1894, 1895, 1896, 1897, 1898, 1899, 1900, 1901, 1902, 1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912, 1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922, 1923, 1924, 1925, 1926, 1927, 1928, 1929, 1930, 1931, 1932, 1933, 1934, 1935, 1936, 1937, 1938, 1939, 1940, 1941, 1942, 1943, 1944, 1945, 1946, 1947, 1948, 1949, 1950, 1951, 1952, 1953, 1954, 1955, 1956, 1957, 1958, 1959, 1960, 1961, 1962, 1963, 1964, 1965, 1966, 1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025, 2026, 2027, 2028, 2029, 2030, 2031, 2032, 2033, 2034, 2035, 2036, 2037, 2038, 2039, 2040, 2041, 2042, 2043, 2044, 2045, 2046, 2047, 2048, 2049, 2050, 2051, 2052, 2053, 2054, 2055, 2056, 2057, 2058, 2059, 2060, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068, 2069, 2070, 2071, 2072, 2073, 2074, 2075, 2076, 2077, 2078, 2079, 2080, 2081, 2082, 2083, 2084, 2085, 2086, 2087, 2088, 2089, 2090, 2091, 2092, 2093, 2094, 2095, 2096, 2097, 2098, 2099, 2100, 2101, 2102, 2103, 2104, 2105, 2106, 2107, 2108, 2109, 2110, 2111, 2112, 2113, 2114, 2115, 2116, 2117, 2118, 2119, 2120, 2121, 2122, 2123, 2124, 2125, 2126, 2127, 2128, 2129, 2130, 2131, 2132, 2133, 2134, 2135, 2136, 2137, 2138, 2139, 2140, 2141, 2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151, 2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161, 2162, 2163, 2164, 2165, 2166, 2167, 2168, 2169, 2170, 2171, 2172, 2173, 2
```

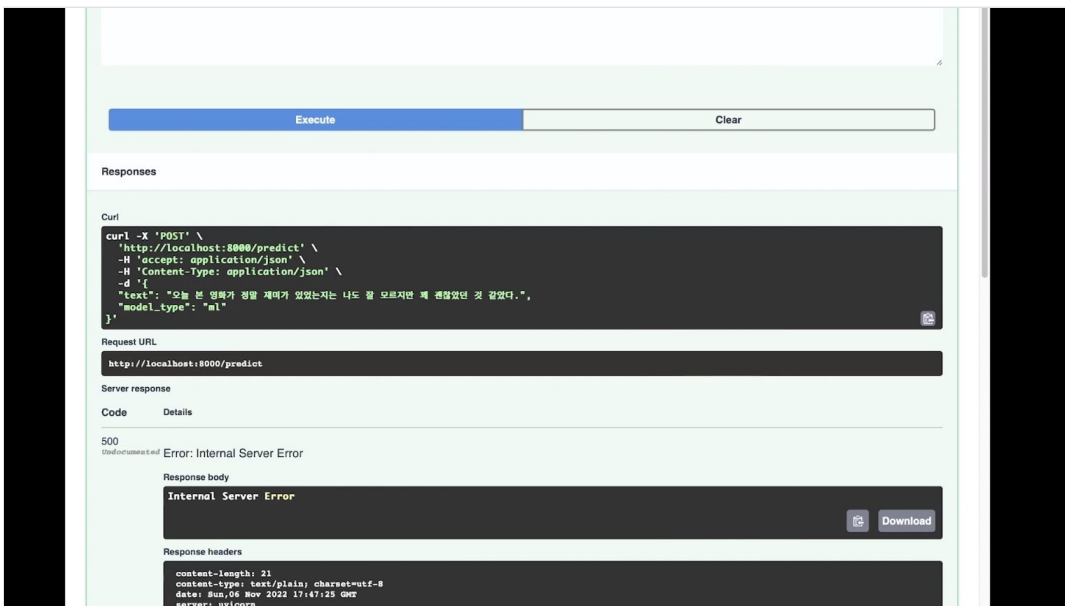
여기에 /docs 라고 입력을 해주면 swagger ui가 나오게 된다. 여기서 우리가 정의했던 predict api를 테스트 해 볼수 있다.



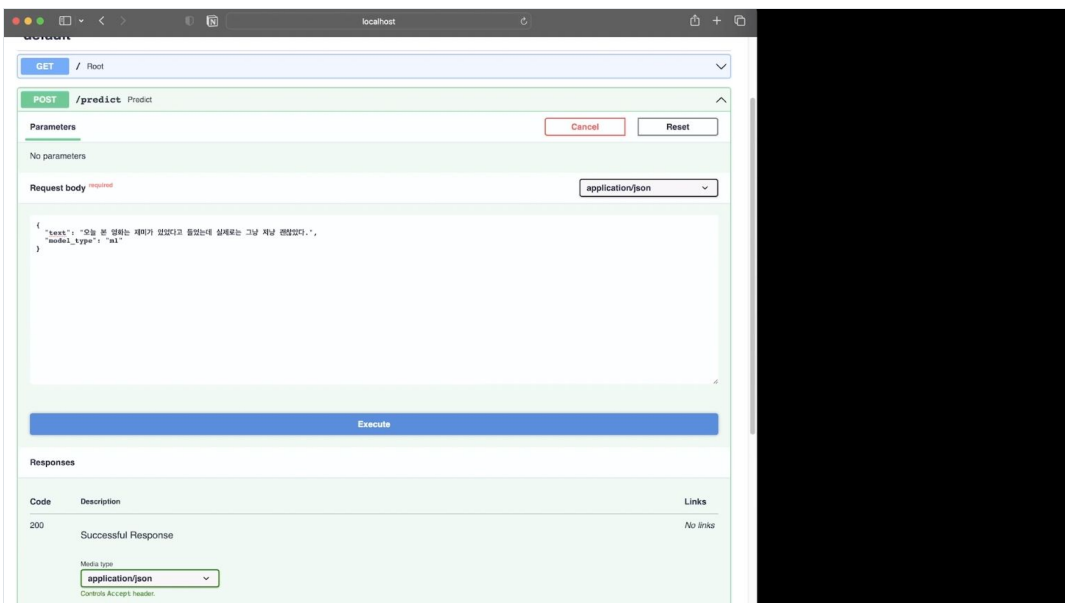
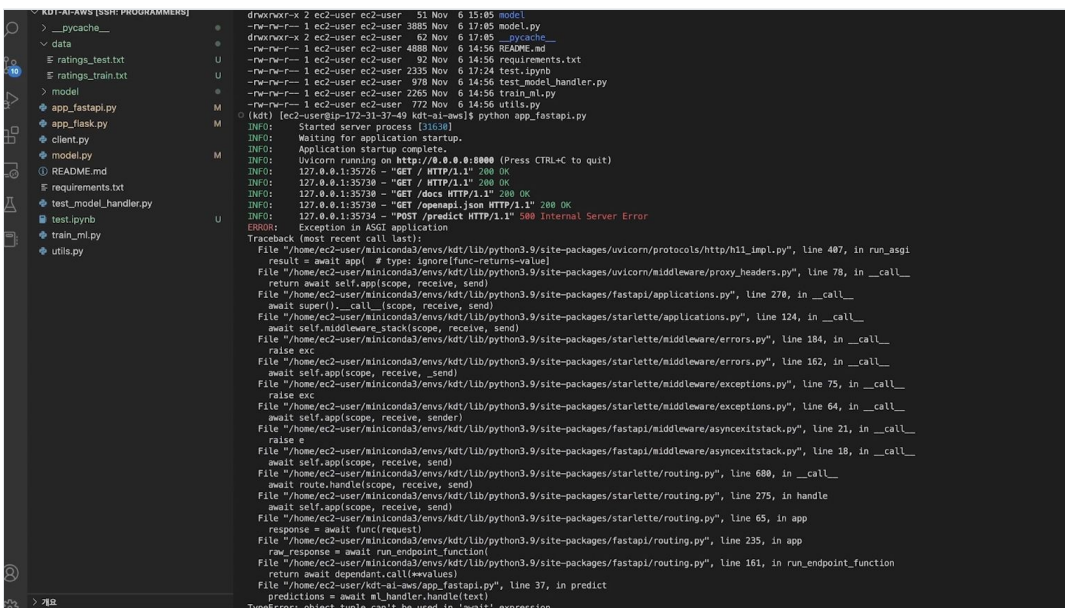
한번 테스트를 진행해보자



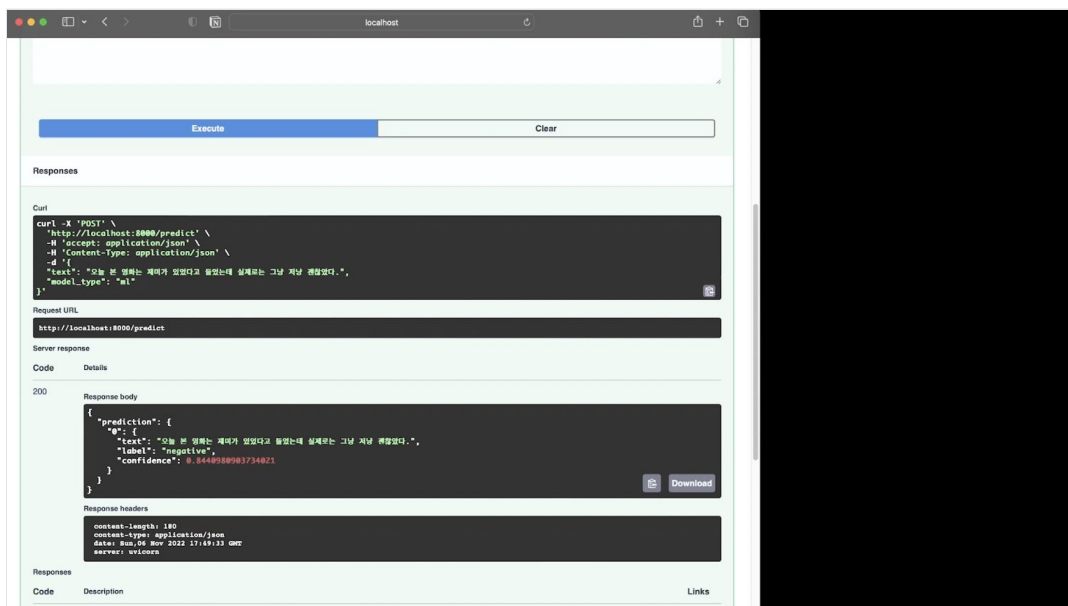
아래 execute 버튼을 누르면 실제 호출이 된다.



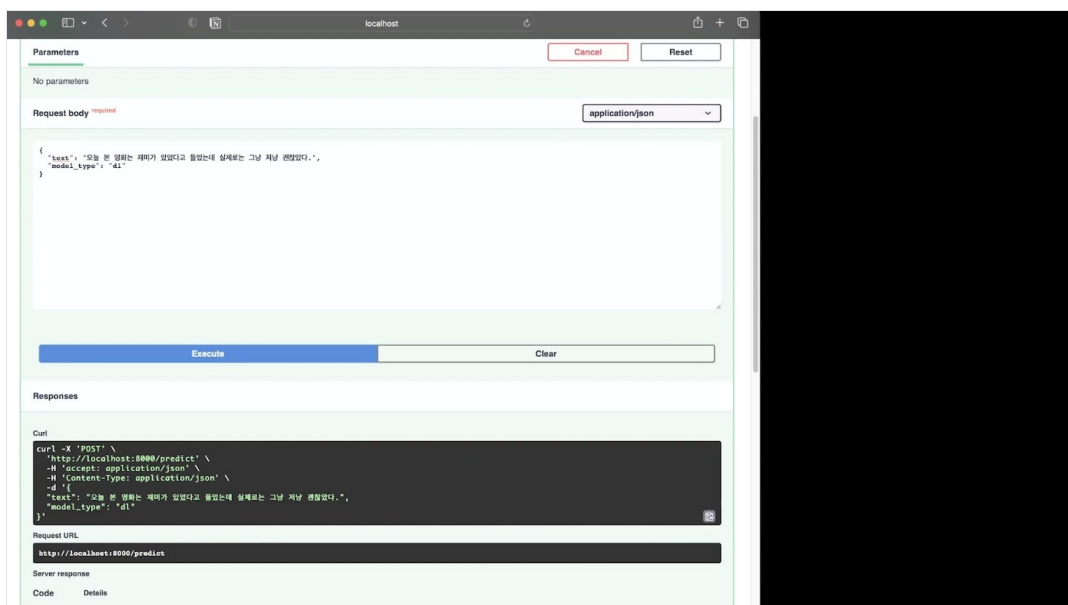
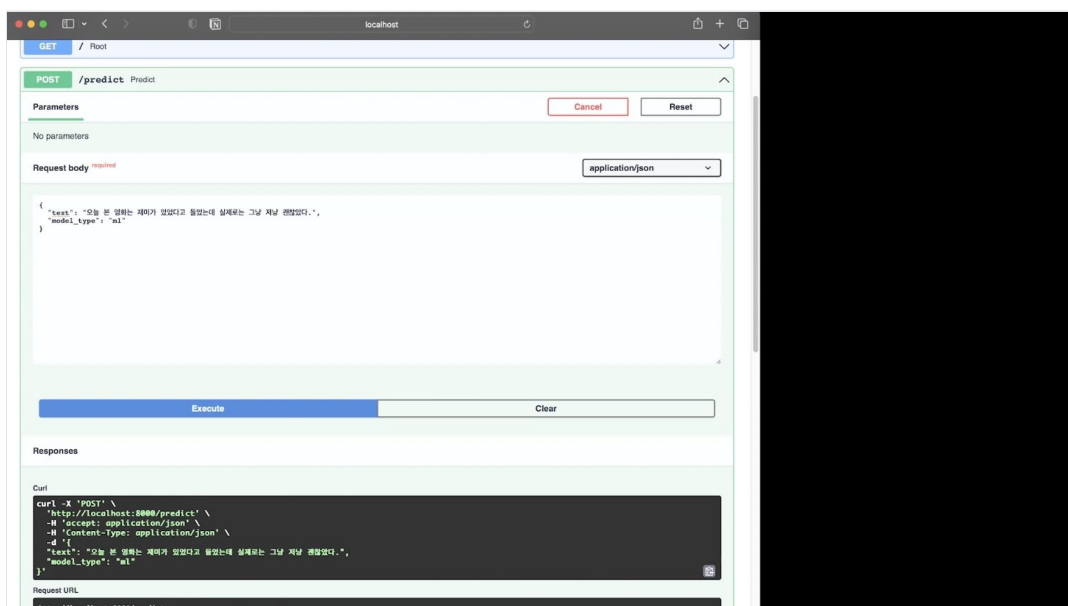
동작했더니 error가 발생한 모습이다. 따라서 app.flask.py에 적은 앞에 await키워드를 지워줘야 한다. 지우고 다시 진행하자!



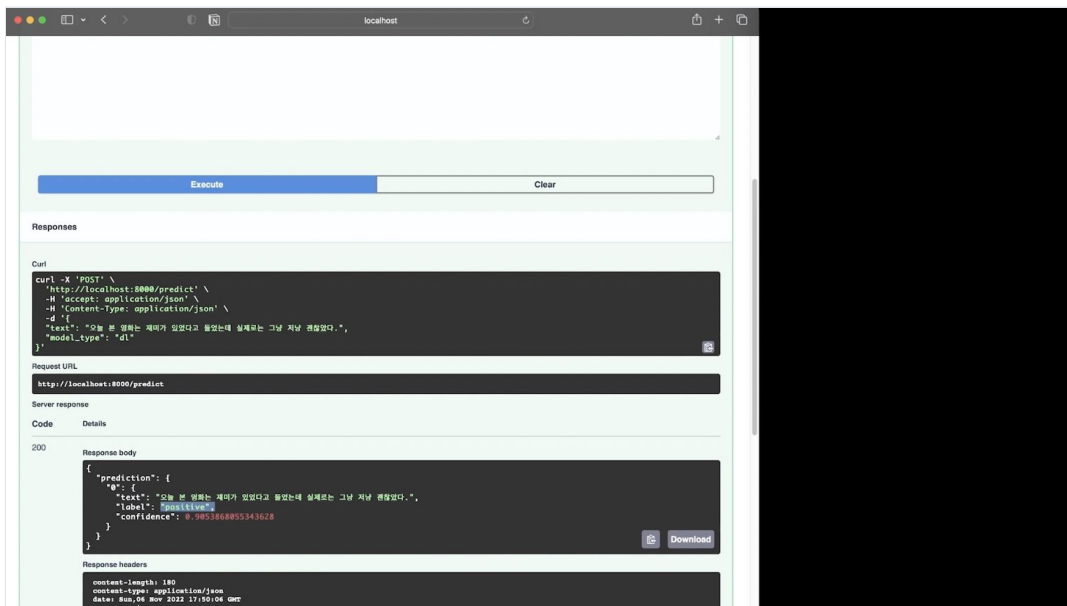
execute를 누르면 밑에 curl에 요청된 내용을 확인해 볼 수있다.



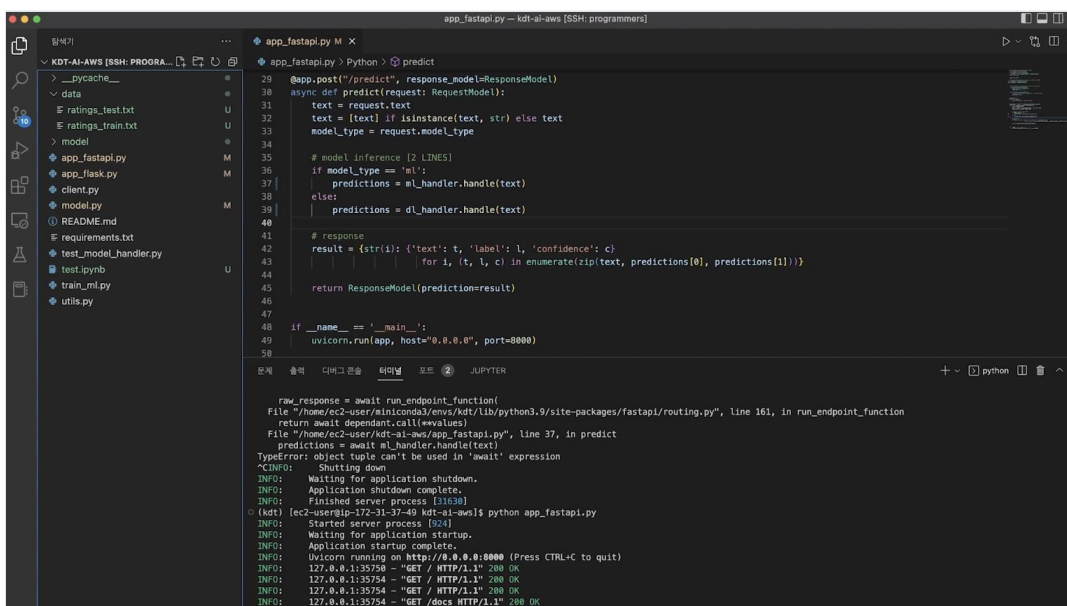
response는 200으로 성공적으로 호출이 완료된 모습, 이 결과가 부정이라고 나왔다. 우리는 이 모델을 딥러닝 모델로 다시 테스트를 해볼 것이다.



딥러닝 모델은 긍정이라고 예측했다 확실히 ml보다 성능이 좋은 것을 확인 해 볼 수있다.



이렇게 swagger ui를 통해서 내가 만든 api를 바로바로 테스트 할 수있다는게 큰 장점이고 디버깅에도 유리하고 편리하다. 터미널도 보면 실제로 docs들을 통해서 호출된 요청들이 보인다.



마찬가지로 flask를 통해 했던것 처럼 test를 해볼 수있다. test.ipynb 에서 테스트를 해보자

8. FastAPI 기반 감성분석 API 개발
103

Test API on remote

Copied

원격에서 서버로 API에 요청하여 테스트 수행

host: EC2 인스턴스 생성 시에 받은 퍼블릭 IP 주소 또는 직접 할당한 탄력적 IP 주소 (ex. 54.83.1.129)

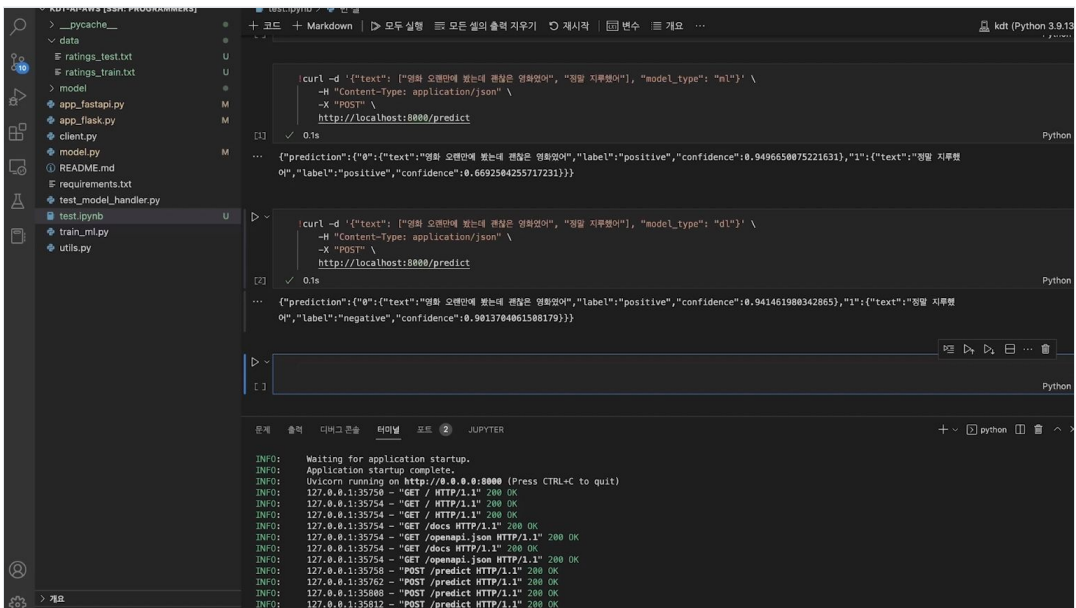
port: EC2 인스턴스 생성 시에 설정했던 port 번호 (ex. 5000)

```
$ curl -d '{"text": ["영화 오렌지에 봤는데 괜찮은 영화였어", "정말 지루했어"], "model_type": "mL"]}' \
-H "Content-Type: application/json" \
-X POST \
http://{HOSTNAME}:{PORT}/predict
```

```
import request

url = f'http://{HOSTNAME}:{PORT}/predict'
data = {"text": ["영화 오렌지에 봤는데 괜찮은 영화였어", "정말 지루했어"], "model_type": "mL"}
response = requests.post(url, json=data)
print(response.content)
```

*You should modify above code
uson=data → json=data



Test API on Swagger UI

브라우저에서 hostname:port/docs 로 접속하여 테스트 수행

