



예측 서비스 배포

5분

실시간 추론용 추론 파이프라인을 만들고 테스트한 후에는 이를 클라이언트 응용 프로그램에서 사용할 서비스로 게시할 수 있습니다.

① 참고

이 연습에서는 ACI(Azure Container Instance)에 웹 서비스를 배포합니다. 이러한 유형의 컴퓨팅은 동적으로 만들어지며 개발 및 테스트에 유용합니다. 프로덕션 환경에서는 '유추 클러스터'를 만들어 향상된 스케일링 성능 및 보안을 제공하는 AKS(Azure Kubernetes Service) 클러스터를 제공해야 합니다.



서비스 배포


- 이전 단원에서 만든 **자동차 가격 예측** 유추 파이프라인을 확인합니다.
- 오른쪽 위에서 **배포**를 선택하고 다음 설정을 사용하여 새로운 실시간 엔드포인트를 배포합니다.
 - **이름**: predict-auto-price
 - **설명**: 자동 가격 회귀
 - **컴퓨팅 형식**: Azure Container Instances
- 웹 서비스가 배포될 때까지 기다립니다. 몇 분 정도 걸릴 수 있습니다. 배포 상태는 디자이너 인터페이스의 왼쪽 상단에 표시됩니다.

서비스 테스트

이제 클라이언트 애플리케이션에서 배포된 서비스를 테스트할 수 있습니다. 여기에서는 아래 셀의 코드를 사용하여 클라이언트 애플리케이션을 시뮬레이션합니다.

- 엔드포인트** 페이지에서 **predict-auto-price** 실시간 엔드포인트를 엽니다.
- predict-auto-price** 엔드포인트가 열리면 **사용** 탭을 보고 다음 정보를 확인합니다. 클라이언트 애플리케이션에서 배포된 서비스로 연결하려면 해당 정보가 필요합니다.
 - 서비스에 대한 REST 엔드포인트
 - 서비스에 대한 기본 키

- 해당 값 옆에 있는  링크를 사용하여 이를 클립보드에 복사할 수 있습니다.
- 브라우저에서 **predict-auto-price** 서비스 페이지의 **사용** 페이지가 열린 상태에서 새 브라우저 탭을 열고 [Azure Machine Learning Studio](#) 의 두 번째 인스턴스를 엽니다. 그런 다음, 새 탭에서 **Notebooks** 페이지(작성자 아래)를 봅니다.
- Notebooks** 페이지의 **내 파일** 에서  단추를 사용하여 다음 설정으로 새 파일을 만듭니다.
 - 파일 위치:** Users/사용자 이름
 - 파일 이름:** Test-Autos
 - 파일 형식:** Notebook
 - 이미 있는 경우 덮어쓰기:** 선택됨
- 새 Notebook이 만들어지면 이전에 만든 컴퓨팅 인스턴스가 **컴퓨팅** 상자에서 선택되었는지, 그리고 상태가 **실행 중** 인지 확인합니다.
- << 단추를 사용하여 파일 탐색기 창을 축소하고 **Test-Autos.ipynb** Notebook 탭에 사용할 수 있는 공간을 확보합니다.
- Notebook에서 만들어진 사각형 셀에 다음 코드를 붙여넣습니다.

Python	 복사
<pre> endpoint = 'YOUR_ENDPOINT' #Replace with your endpoint key = 'YOUR_KEY' #Replace with your key import urllib.request import json import os # Prepare the input data data = { "Inputs": { "WebServiceInput0": [{ 'symboling': 3, 'normalized-losses': None, 'make': "alfa-romero", 'fuel-type': "gas", 'aspiration': "std", 'num-of-doors': "two", 'body-style': "convertible", 'drive-wheels': "rwd", 'engine-location': "front", 'wheel-base': 88.6, 'length': 168.8, 'width': 64.1, 'height': 48.8, 'curb-weight': 2548, 'engine-type': "dohc", </pre>	

```

        'num-of-cylinders': "four",
        'engine-size': 130,
        'fuel-system': "mpfi",
        'bore': 3.47,
        'stroke': 2.68,
        'compression-ratio': 9,
        'horsepower': 111,
        'peak-rpm': 5000,
        'city-mpg': 21,
        'highway-mpg': 27,
    },
],
},
"GlobalParameters": {
}
}
body = str.encode(json.dumps(data))
headers = {'Content-Type': 'application/json', 'Authorization': ('Bearer ' +
key)}
req = urllib.request.Request(endpoint, body, headers)

try:
    response = urllib.request.urlopen(req)
    result = response.read()
    json_result = json.loads(result)
    y = json_result["Results"]["WebServiceOutput0"][0]["predicted_price"]
    print('Predicted price: {:.2f}'.format(y))

except urllib.error.HTTPError as error:
    print("The request failed with status code: " + str(error.code))

    # Print the headers to help debug the error
    print(error.info())
    print(json.loads(error.read().decode("utf8", 'ignore')))
```

❗ 참고

코드의 세부 정보에 대해 너무 걱정하지 마세요. 자동차의 세부 정보를 제출하고 만들어진 **predict-auto-price** 서비스를 사용하여 예측 가격을 얻습니다.

9. **predict-auto-price** 서비스의 **사용** 페이지가 포함된 브라우저 탭으로 전환한 다음 서비스에 대한 REST 엔드포인트를 복사합니다. Notebook을 포함하는 탭으로 돌아가 키를 코드에 복사하고 YOUR_ENDPOINT를 대체합니다.
10. **predict-auto-price** 서비스의 **사용** 페이지가 포함된 브라우저 탭으로 전환한 다음 서비스에 대한 기본 키를 복사합니다. Notebook을 포함하는 탭으로 돌아가 키를 코드에 복사하고 YOUR_KEY를 대체합니다.
11. Notebook을 저장합니다. 셀 옆에 있는 ▶ 단추를 사용하여 코드를 실행합니다.