Training and Deploying TensorFlow Models at Scale [2]
– Google Cloud AI Platform

Samkeun Kim <skim@hknu.ac.kr>

http://cyber.hknu.ac.kr/

Creating a Prediction Service on Google Cloud AI Platform

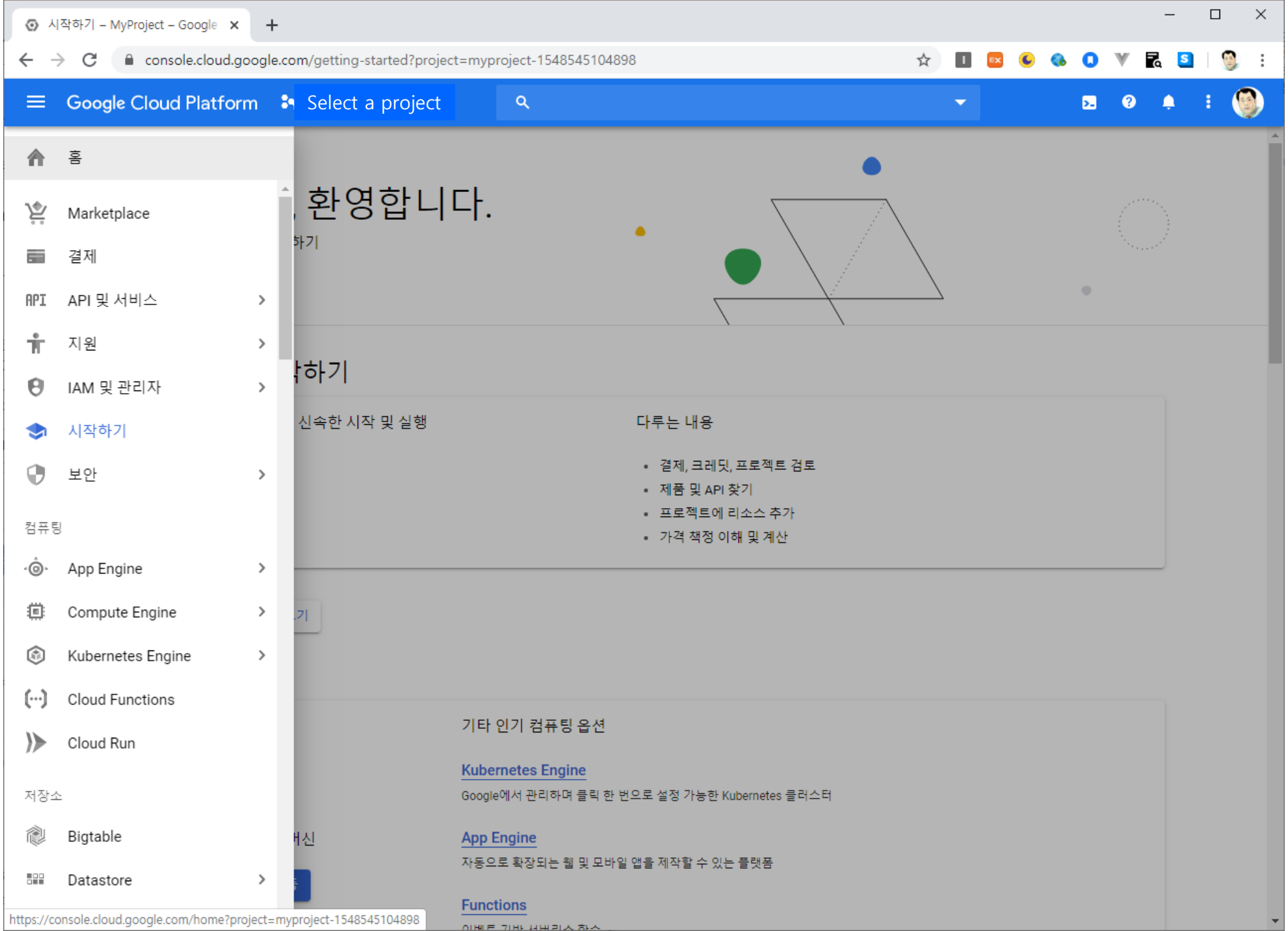
1. Google 계정에 로그인 => Go to the Google Cloud Platform (GCP) console
2. GCP를 처음 사용하는 경우 이용 약관을 읽고 동의해야 한다.

   신규 사용자에게는 12개월 동안 사용할 수 있는 300달러 상당의 GCP 크레딧을 포함한 무료 평가판이 제공된다.

   무료 평가판에 가입 한 후에도 결제 프로필을 작성하고 신용 카드 번호를 입력해야 한다.

   신용 카드 번호는 확인 목적으로 사용되며 (무료 평가판을 여러 번 사용하는 사람들을 피하기 위해) 사용되지만 청구되지는 않는다.

   요청되면 계정을 활성화하고 업그레이드하자.


시작하기 – MyProject – Google
console.cloud.google.com/getting-started?project=myproject-1548545104898
Google Cloud Platform
Select a project
홈
Marketplace
결제
API 및 서비스
지원
IAM 및 관리자
시작하기
보안
컴퓨팅
App Engine
Compute Engine
Kubernetes Engine
Cloud Functions
Cloud Run
저장소
Bigtable
Datastore
환영합니다.
하기
신속한 시작 및 실행
다루는 내용
결제, 크레딧, 프로젝트 검토
제품 및 API 찾기
프로젝트에 리소스 추가
가격 책정 이해 및 계산
기타 인기 컴퓨팅 옵션
Kubernetes Engine
Google에서 관리하며 클릭 한 번으로 설정 가능한 Kubernetes 클러스터
App Engine
자동으로 확장되는 웹 및 모바일 앱을 제작할 수 있는 플랫폼
Functions
https://console.cloud.google.com/home?project=myproject-1548545104898

3. 이전에 GCP를 사용한 적이 있고 무료 평가판이 만료된 경우 이 장에서 사용할 서비스에 약간의 비용이 든다.

   더 이상 필요하지 않을 때 서비스를 끄는 것을 잊지 말자.

   서비스를 실행하기 전에 가격 조건을 이해하고 동의해야 한다.

   또한 결제 계정이 활성화되어 있는지 확인하자:

   확인하려면 왼쪽의 탐색 메뉴를 열고 결제를 클릭하고 결제 방법을 설정하고 결제 계정이 활성화되어 있는지 확인하면 된다.

4. GCP의 모든 리소스는 프로젝트에 속한다.

   여기에는 사용 가능한 모든 가상 머신, 저장한 파일 및 실행중인 작업이 포함된다.

   계정을 만들면 GCP가 'My First Project'라는 프로젝트를 자동으로 생성한다.

   원하는 경우 화면 왼쪽의 탐색 메뉴에서 프로젝트 설정으로 이동하여 표시 이름을 변경할 수 있다:

   IAM 및 관리자 → 설정을 선택하고 프로젝트의 표시 이름을 변경한 후 저장을 클릭하면 된다.

   프로젝트에는 고유한 ID와 번호도 있다. 프로젝트를 만들 때 프로젝트 ID를 선택할 수 있지만 나중에 변경할 수는 없다.

   프로젝트 번호는 자동으로 생성되며 변경할 수 없다.

   새 프로젝트를 작성하려면 페이지 맨 위에 있는 프로젝트 이름을 클릭한 다음 새 프로젝트를 클릭하고 프로젝트 ID를 입력하면 된다.

   이 새 프로젝트에 대해 Billing이 활성화되어 있는지 확인한다.

5. 이제 결제가 활성화된 GCP 계정이 있으므로 서비스 사용을 시작할 수 있다.

   가장 먼저 필요한 것은 Google Cloud Storage(GCS)이다: 여기에 저장된 모델, 학습 데이터 등을 넣을 수 있다.

   탐색 메뉴에서 **저장소** 섹션으로 아래로 스크롤하고 **Storage** → **브라우저**를 클릭한다.

   모든 파일은 하나 이상의 버킷에 들어간다: **버킷 생성**을 클릭하고 버킷 이름을 입력한다.

   GCS는 버킷에 전 세계적으로 단일 네임 스페이스를 사용하므로 "machine-learning"과 같은 간단한 이름을 사용할 수 없다.

   버킷 이름은 DNS 레코드에 사용될 수 있으므로 DNS 이름 지정 규칙을 준수해야 한다.

   또한 버킷 이름은 공개이므로 비공개 항목을 넣지 않아야 한다.

   고유성을 보장하기 위해 접두사로 도메인 이름 또는 회사 이름을 사용하거나 이름의 일부분으로 임의의 숫자를 사용하는 것이 일반적이다.

   버킷을 호스팅할 위치를 선택하면 나머지 옵션들은 디폴트로 넣어진다.

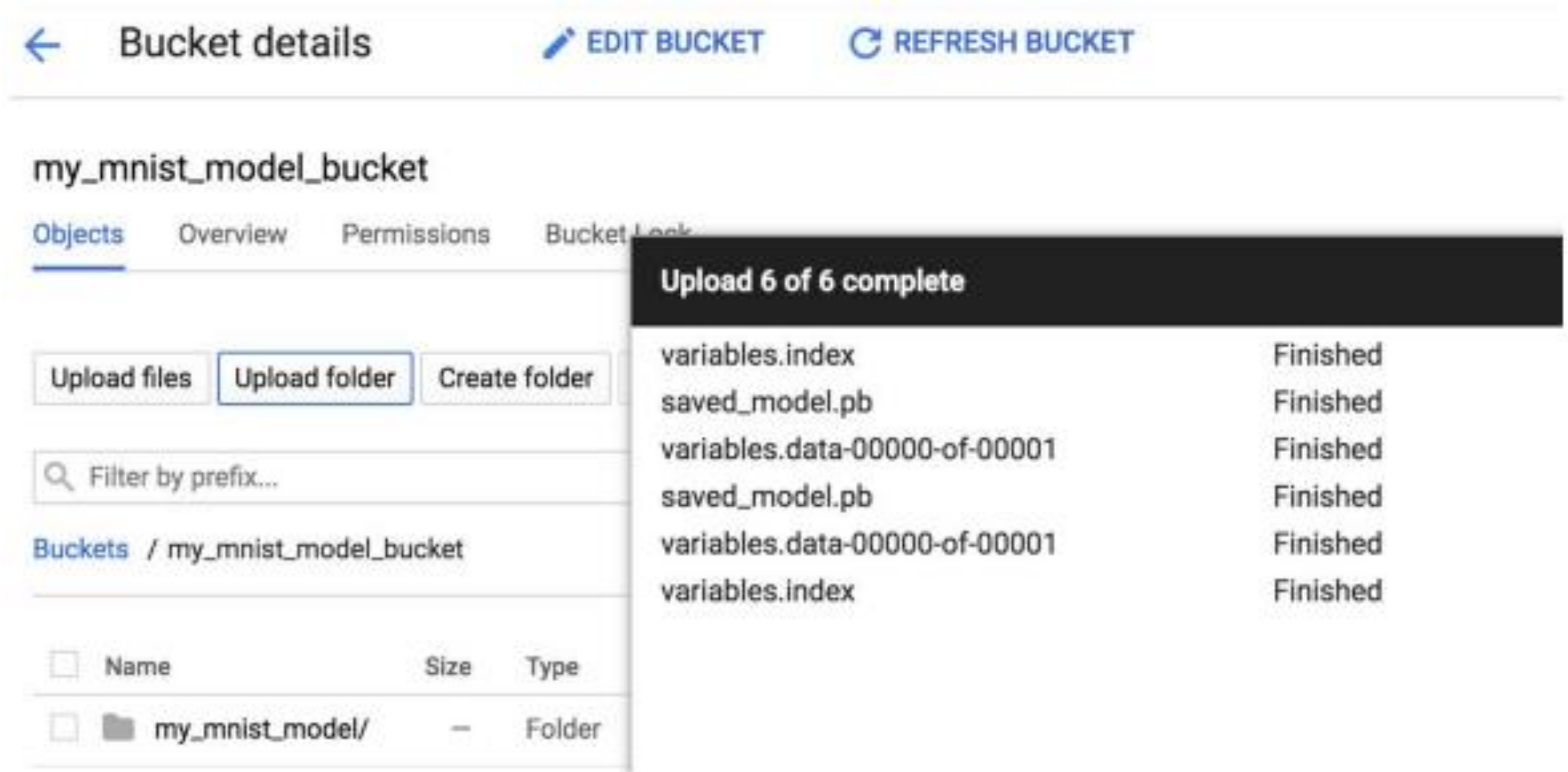
   마지막으로 **만들기**를 클릭한다.

6. 이전에 생성한 *my_mnist_model* 폴더(하나 이상의 버전 포함)를 버킷에 업로드한다.

   이렇게 하려면 GCS 브라우저로 이동하여 버킷을 클릭한 다음 *my_mnist_model* 폴더를 시스템에서 버킷으로 드래그 앤 드롭한다.

   또는"Upload folder(폴더 업로드)"를 클릭하고 업로드 할 *my_mnist_model* 폴더를 선택할 수 있다.

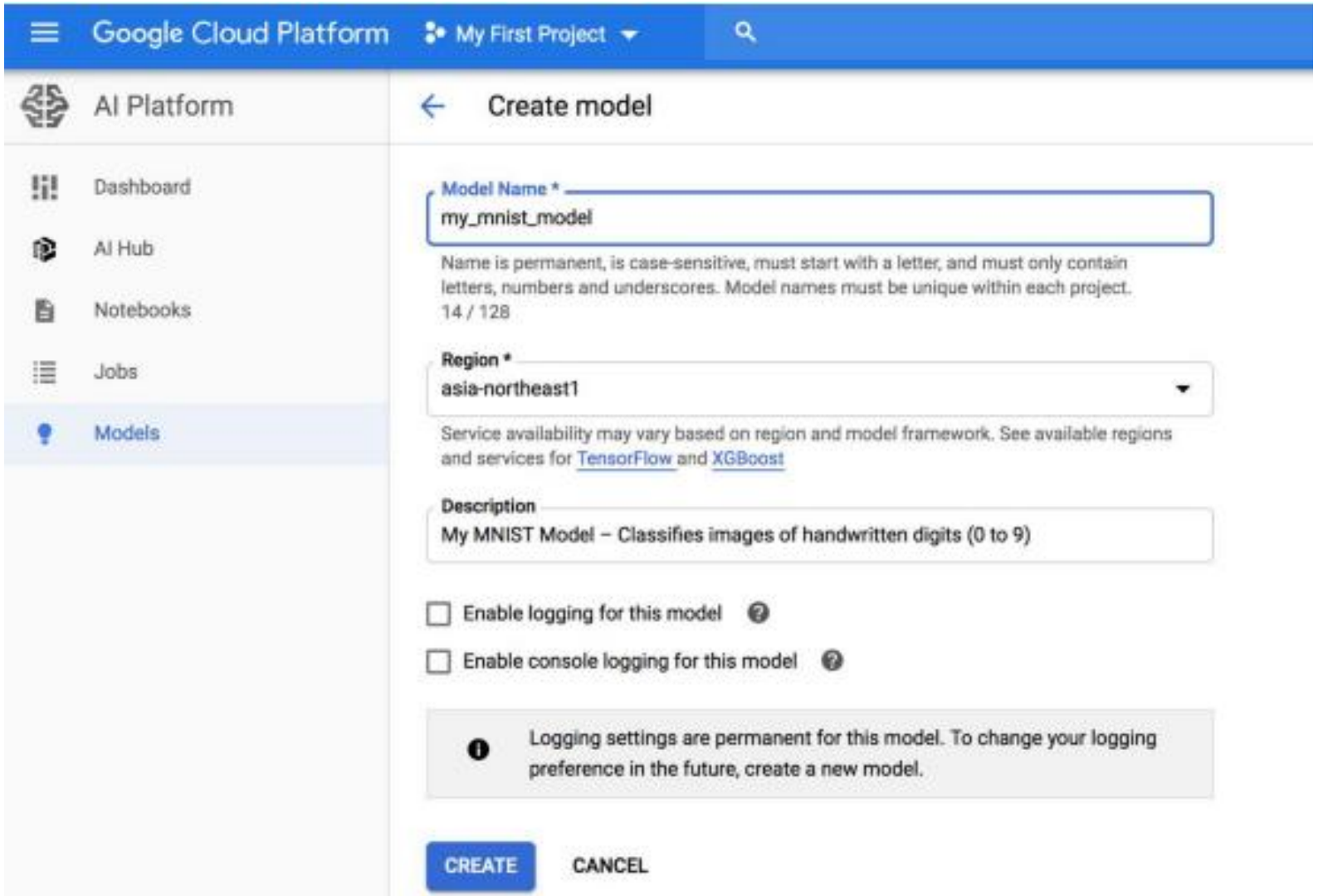
   기본적으로 SavedModel의 최대 크기는 250 MB이지만 더 높은 할당량을 요청할 수도 있다.


Bucket details
EDIT BUCKET
REFRESH BUCKET
my_mnist_model_bucket
Objects
Overview
Permissions
Upload files
Upload folder
Create folder
Filter by prefix...
Buckets / my_mnist_model_bucket
Name
Size
Type
my_mnist_model/
—
Folder
Upload 6 of 6 complete
variables.index
Finished
saved_model.pb
Finished
variables.data-00000-of-00001
Finished
saved_model.pb
Finished
variables.data-00000-of-00001
Finished
variables.index
Finished

7. 이제 사용할 모델과 버전을 알 수 있도록 AI Platform(이전의 ML 엔진)을 설정해야 한다.

   탐색 메뉴에서 **인공지능** 섹션으로 아래로 스크롤하고 **AI Platform** → **모델**을 클릭한다.

   새 모델(몇 분 소요)을 클릭해서 모델 세부 정보를 입력하고 **만들기**를 클릭한다.

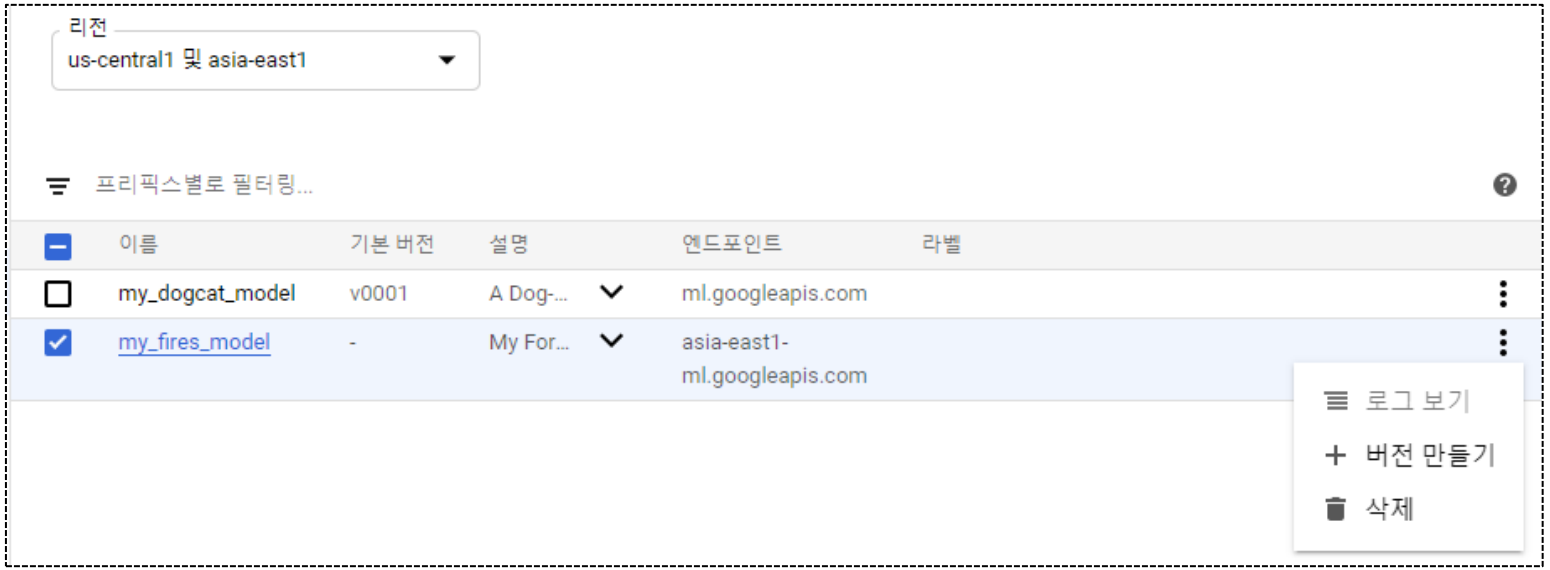

Google Cloud Platform
My First Project
AI Platform
Create model
Dashboard
AI Hub
Notebooks
Jobs
Models
Model Name *
my_mnist_model
Name is permanent, is case-sensitive, must start with a letter, and must only contain letters, numbers and underscores. Model names must be unique within each project.
14 / 128
Region *
asia-northeast1
Service availability may vary based on region and model framework. See available regions and services for TensorFlow and XGBoost
Description
My MNIST Model – Classifies images of handwritten digits (0 to 9)
Enable logging for this model
Enable console logging for this model
Logging settings are permanent for this model. To change your logging preference in the future, create a new model.
CREATE
CANCEL

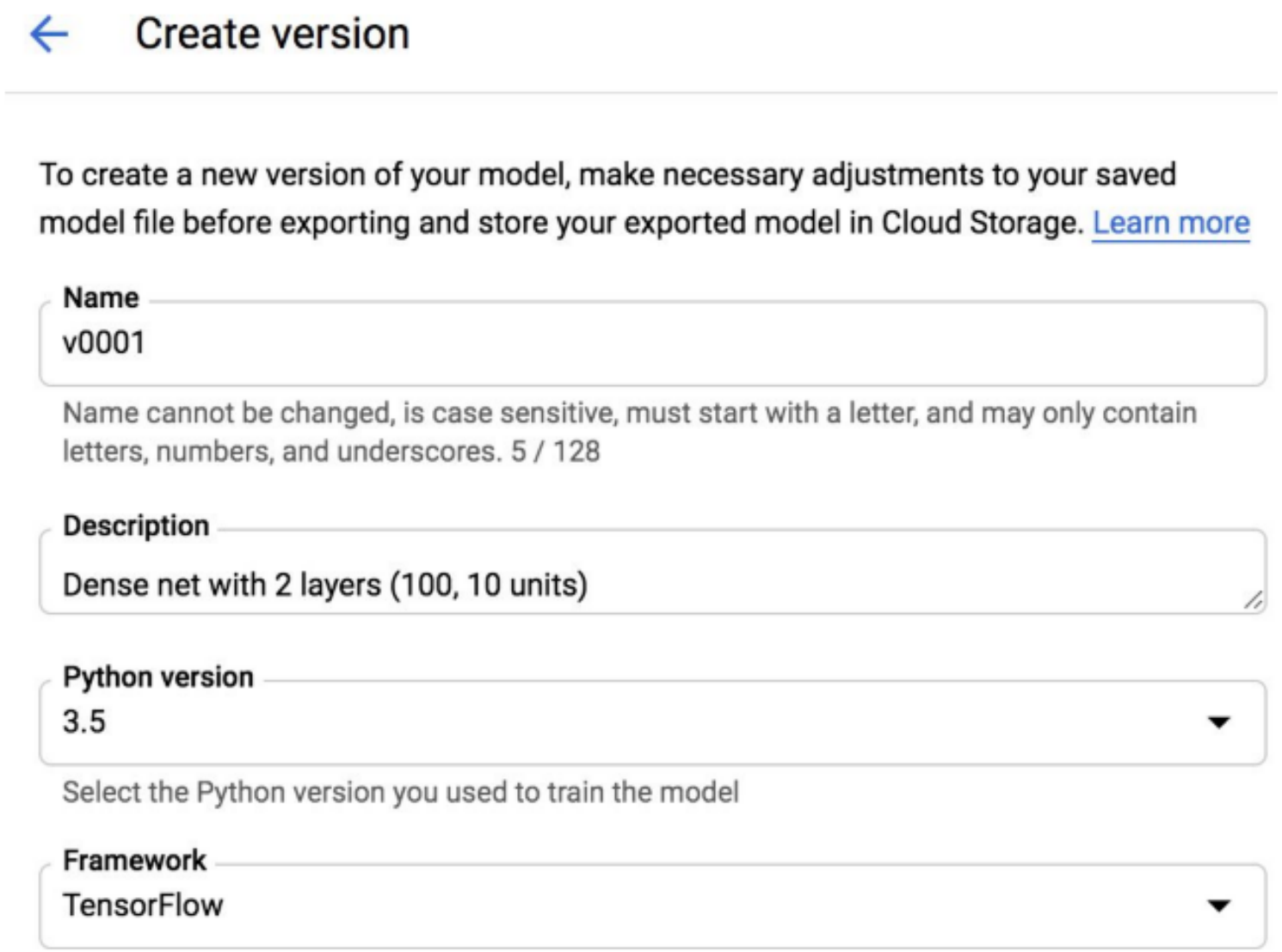
8. AI Platform에 모델을 만들었으므로 모델 버전을 작성해야 한다.

   모델 목록에서 방금 생성한 모델을 클릭한 다음 "버전 만들기"를 클릭하고 버전 세부 정보를 입력한다:

   이름, 설명, Python 버전(3.5 이상),
   프레임 워크(TensorFlow) 설정,
   프레임 워크 버전(사용 가능한 경우 2.0 또는 1.13),
   ML 런타임 버전(사용 가능한 경우 2.0 또는 1.13),
   머신 유형(현재 "단일 코어 CPU"선택),
   GCS의 모델 경로(이는 실제 버전 폴더에 대한 전체 경로,예를 들어, `gs://my-mnist-model-bucket/my_mnist_model/0001/`),
   스케일링(자동 선택) 및 항상 실행중인 최소 TF Serving 컨테이너 수를 비워둬야 한다. (그렇지 않으면 많은 요금이 부과될 것이다.)

   그런 다음 **저장**을 클릭한다.




Create version
To create a new version of your model, make necessary adjustments to your saved model file before exporting and store your exported model in Cloud Storage. Learn more
Name
v0001
Name cannot be changed, is case sensitive, must start with a letter, and may only contain letters, numbers, and underscores. 5 / 128
Description
Dense net with 2 layers (100, 10 units)
Python version
3.5
Select the Python version you used to train the model
Framework
TensorFlow

NOTE:

스케일링(자동 선택) 및 항상 실행중인 최소 TF Serving 컨테이너 수를 비워 둔 경우:

AI Platform은 예측 서비스를 사용하지 않으면 모든 컨테이너를 중지한다. 즉, 사용하는 스토리지 양(월별 기가 바이트 당 몇 센트)에 대해서만 비용을 지불하면 된다.

따라서 사용자가 서비스에 쿼리할 때만 TF Serving 컨테이너를 시작하게 하는데 몇 초가 걸린다.

이런 지연시간을 피하고 싶다면 모델 버전을 작성할 때 최소 TF Serving 컨테이너 수를 1로 설정하면 된다.

물론 이렇게 설정하면 적어도 하나의 머신이 지속적으로 실행되므로 월별 요금이 조금 높아진다.

Using the Prediction Service

기본적으로 AI Platform은 TF Serving만 실행하므로 원칙적으로 쿼리할 URL을 알고 있다면 이전과 동일한 코드를 사용할 수 있다.

단지 한 가지 문제가 있다: GCP는 암호화 및 인증도 처리한다.

암호화는 SSL/TLS를 기반으로 하며 인증은 토큰을 기반으로 한다:

- 모든 요청에서 비밀 인증 토큰을 서버로 보내야한다.

따라서 코드에서 예측 서비스(또는 다른 GCP 서비스)를 사용하려면 **토큰**을 얻어야 한다.

먼저 인증을 설정하고 애플리케이션에 GCP에 대한 적절한 액세스 권한을 부여해야 한다:

- 클라이언트 코드는 서비스 계정으로 인증할 수 있다.
- 이것은 사용자가 아니라 애플리케이션을 의미하는 계정이다.
- 일반적으로 매우 제한된 액세스 권한이 부여된다.

애플리케이션에 대한 서비스 계정을 만들어 보자:

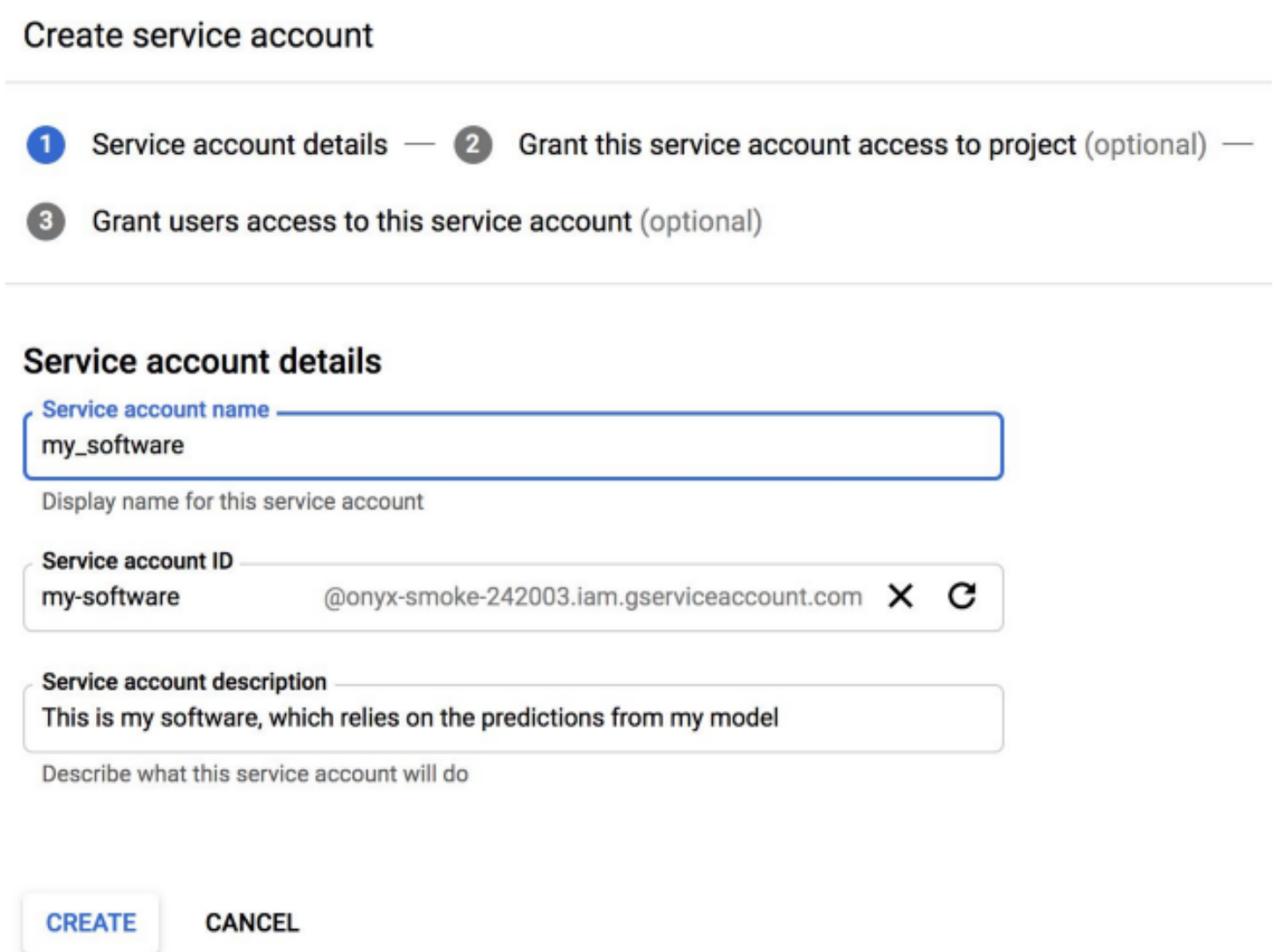
탐색 메뉴에서 **IAM 및 관리자** → **서비스 계정**으로 이동한 다음 **서비스 계정** 생성을 클릭하고 폼의 세부사항(서비스 계정 이름, ID, 설명)을 작성하고 **만들기**를 클릭한다.

다음으로 이 계정에 몇 가지 액세스 권한을 부여해야 한다.

ML 엔진 개발자 역할을 선택한다. 이렇게 하면 서비스 계정이 예측을 수행할 수 있게 된다.

그런 다음 **키 만들기**를 클릭하여 서비스 계정의 개인 키를 내보내고 JSON을 선택한 다음 **만들기**를 클릭한다.

이렇게 하면 개인 키를 JSON 파일 형식으로 다운로드 할 것이다. 개인 키를 잘 보관하자!!


Create service account
1 Service account details — 2 Grant this service account access to project (optional) —
3 Grant users access to this service account (optional)
Service account details
Service account name
my_software
Display name for this service account
Service account ID
my-software
@onyx-smoke-242003.iam.gserviceaccount.com
Service account description
This is my software, which relies on the predictions from my model
Describe what this service account will do
CREATE
CANCEL

이제 예측 서비스를 쿼리하는 스크립트를 작성해 보자:

- Google => 서비스에 대해 쉽게 액세스하게 해주는 라이브러리 제공
- Google API Client Library:

```
$ pip install google-api-python-client==1.7.9
```

- 서비스 계정의 개인 키를 `GOOGLE_APPLICATION_CREDENTIALS` 환경변수에 설정:

```
import os

os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "my_service_account_key.json"
```

예측 서비스에 대해 액세스를 감싸는 리소스 객체 생성:

```
import googleapiclient.discovery

project_id = "onyx-smoke-242003" # change this to your project ID
model_id = "my_mnist_model"
model_path = "projects/{}/models/{}".format(project_id, model_id)
ml_resource = googleapiclient.discovery.build("ml", "v1").projects()
```

리소스 객체를 사용하여 예측 서비스를 호출하는 함수 생성:

```
def predict(X):
    input_data_json = {"signature_name": "serving_default",
                       "instances": X.tolist()}
    request = ml_resource.predict(name=model_path, body=input_data_json)
    response = request.execute()
    if "error" in response:
        raise RuntimeError(response["error"])
    return np.array([pred[output_name] for pred in response["predictions"]])
```

Let's see if it works:

```
>>> Y_probas = predict(X_new)
>>> np.round(Y_probas, 2)
array([[0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 0.  , 1.  , 0.  , 0.  ],
       [0.  , 0.  , 0.99, 0.01, 0.  , 0.  , 0.  , 0.  , 0.  , 0.  ],
       [0.  , 0.96, 0.01, 0.  , 0.  , 0.  , 0.  , 0.01, 0.01, 0.  ]])
```

실습과제 15-1

본문에 나오는 전체 내용 PyCharm에서 실행하기

★ **반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.**

참고: 제 15강 실습과제 15-1 (cloud-client) Training and Deploying TensorFlow Models at Scale [2] - GCP.pdf

당뇨병 진단 예측 프로그램

우리 팀은 의사가 의료 실험실 결과에 따라 여성 환자에 대한 당뇨병 진단을 예측하는 데 도움이 되는 웹 앱을 개발해야 한다.

이 파일에 대하여 (diabetes.csv)

이 데이터 세트는 원래 National Institute of Diabetes and Digestive and Kidney Diseases에서 가져온 것이다. 데이터 세트의 목적은 데이터 세트에 포함된 특정 진단 측정 값을 기반으로 환자에게 당뇨병이 있는지 여부를 진단적으로 예측하는 것이다. 더 큰 데이터베이스에서 몇 가지 제약을 주어서 인스턴스들을 선택했다. 특히, 여기의 모든 환자는 21세 이상의 피마 인디언 여성이다.

데이터 세트는 여러 의료 예측 변수(독립) 변수와 하나의 타깃(종속) 변수인 결과로 구성된다. 독립적인 변수에는 환자의 임신 횟수, BMI, 인슐린 수치, 연령 등이 포함되어 있다.

Columns:

- PregnanciesNumber of times pregnant
- GlucosePlasma glucose concentration a 2 hours in an oral glucose tolerance test
- BloodPressureDiastolic blood pressure (mm Hg)
- SkinThicknessTriceps skin fold thickness (mm)
- Insulin2-Hour serum insulin (mu U/ml)
- BMIBody mass index (weight in kg/(height in m)^2)
- DiabetesPedigreeFunctionDiabetes pedigree function
- AgeAge (years)
- OutcomeClass variable (0 or 1) 268 of 768 are 1, the others are 0

모델 개발

```
# create the model
inputs = keras.Input(shape=(8,))
hidden1 = Dense(12,activation='relu')(inputs)
hidden2 = Dense(8,activation='relu',)(hidden1)
output = Dense(1,activation='sigmoid')(hidden2)
model = keras.Model(inputs,output)

# summary to verify the structure
model.summary()

# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])

# train the model on the training data
history = model.fit(X_train,y_train,epochs=100,batch_size=16,verbose=0)
```

```
# create the variables to save the model
model_version = "0002"
model_name = "my_pima_model"
model_path = os.path.join(model_name, model_version)
print("\nmodel_path: \n",
      model_path)

# save the model
tf.saved_model.save(model, model_path)
```

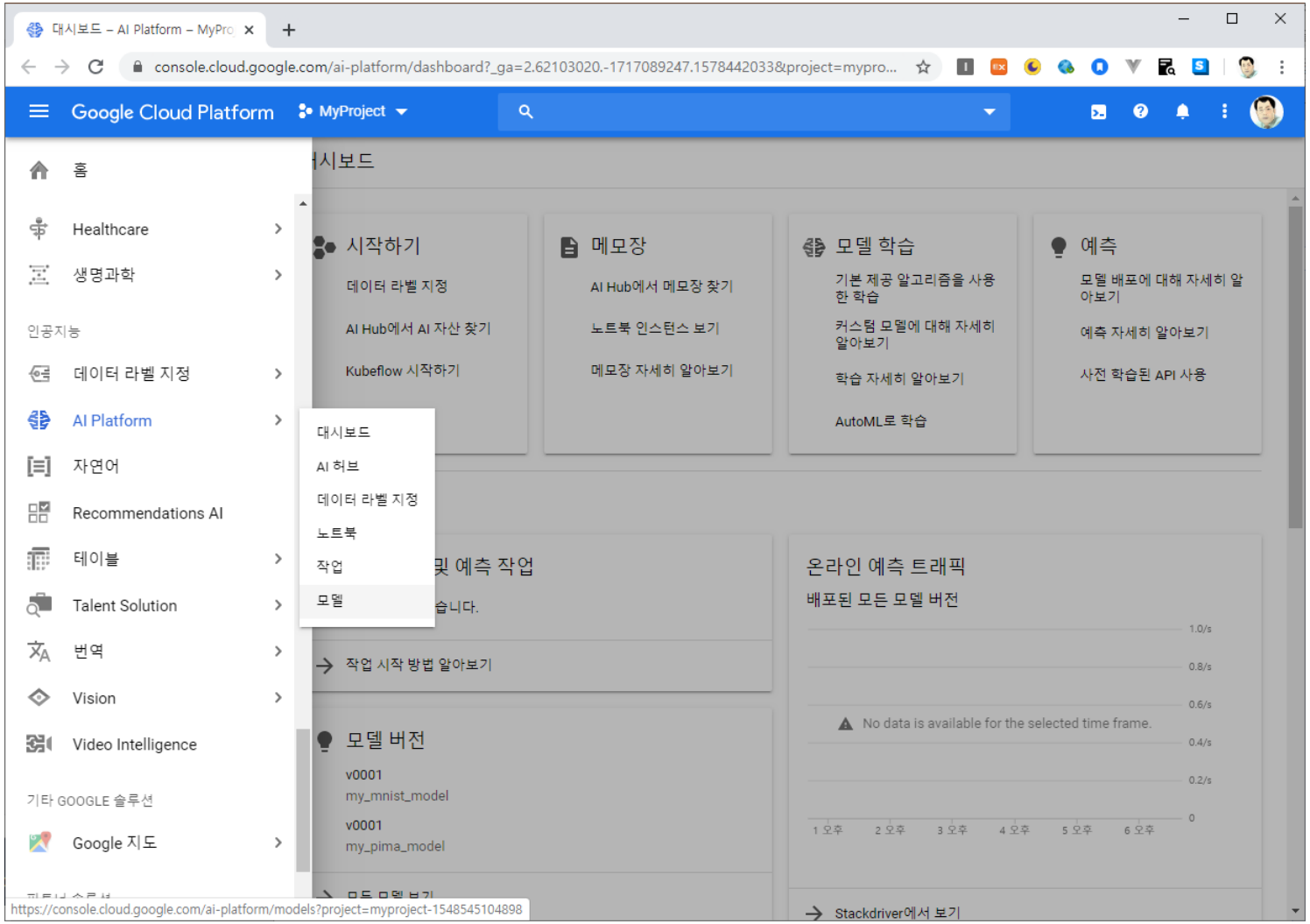
실습과제 15-2

본문에 나오는 전체 내용 PyCharm에서 실행하기

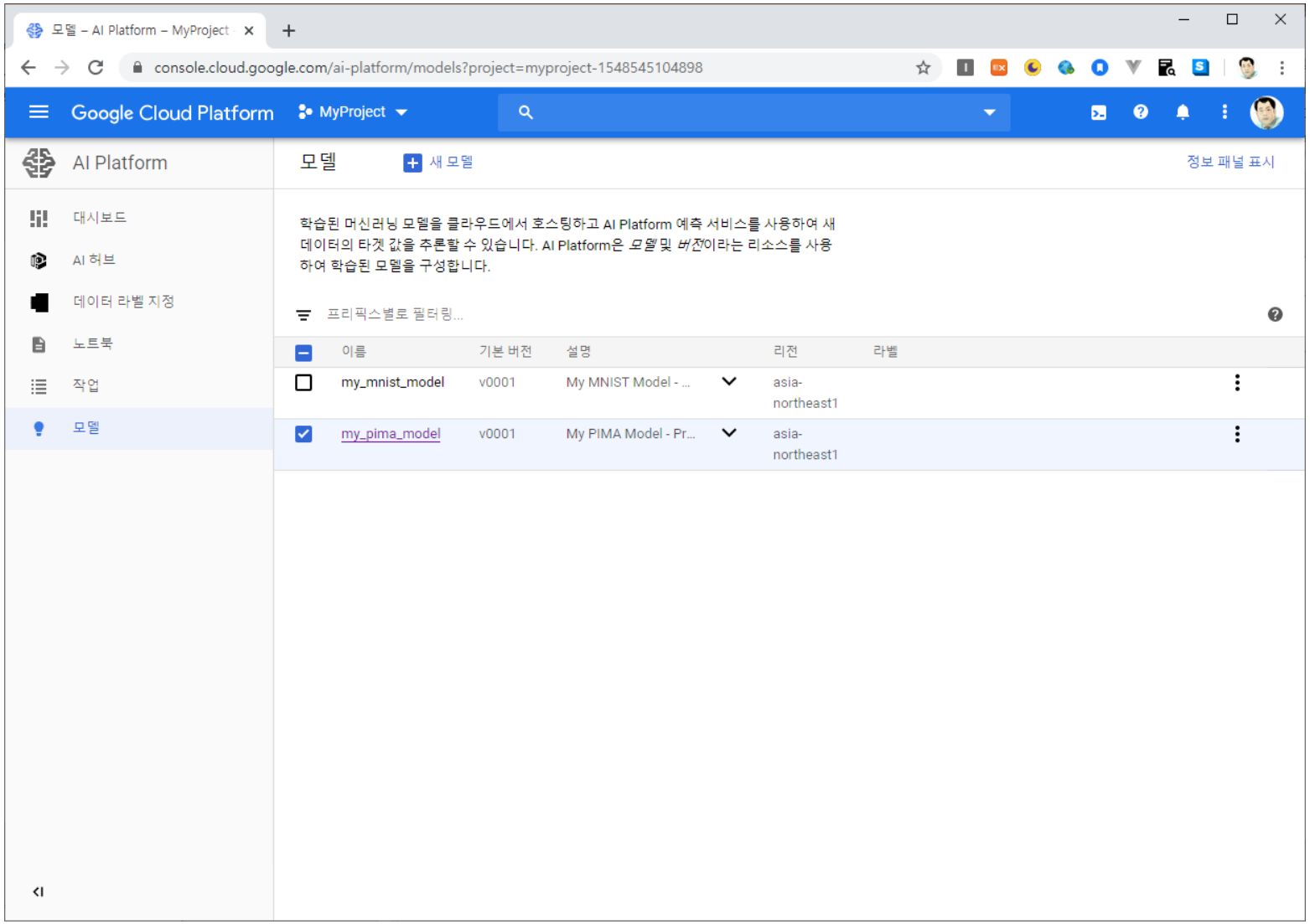
★ **반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.**

참고: 제 15강 실습과제 15-2 (pima-model) How to build and deploy a diabetes diagnostic app - GCP.pdf

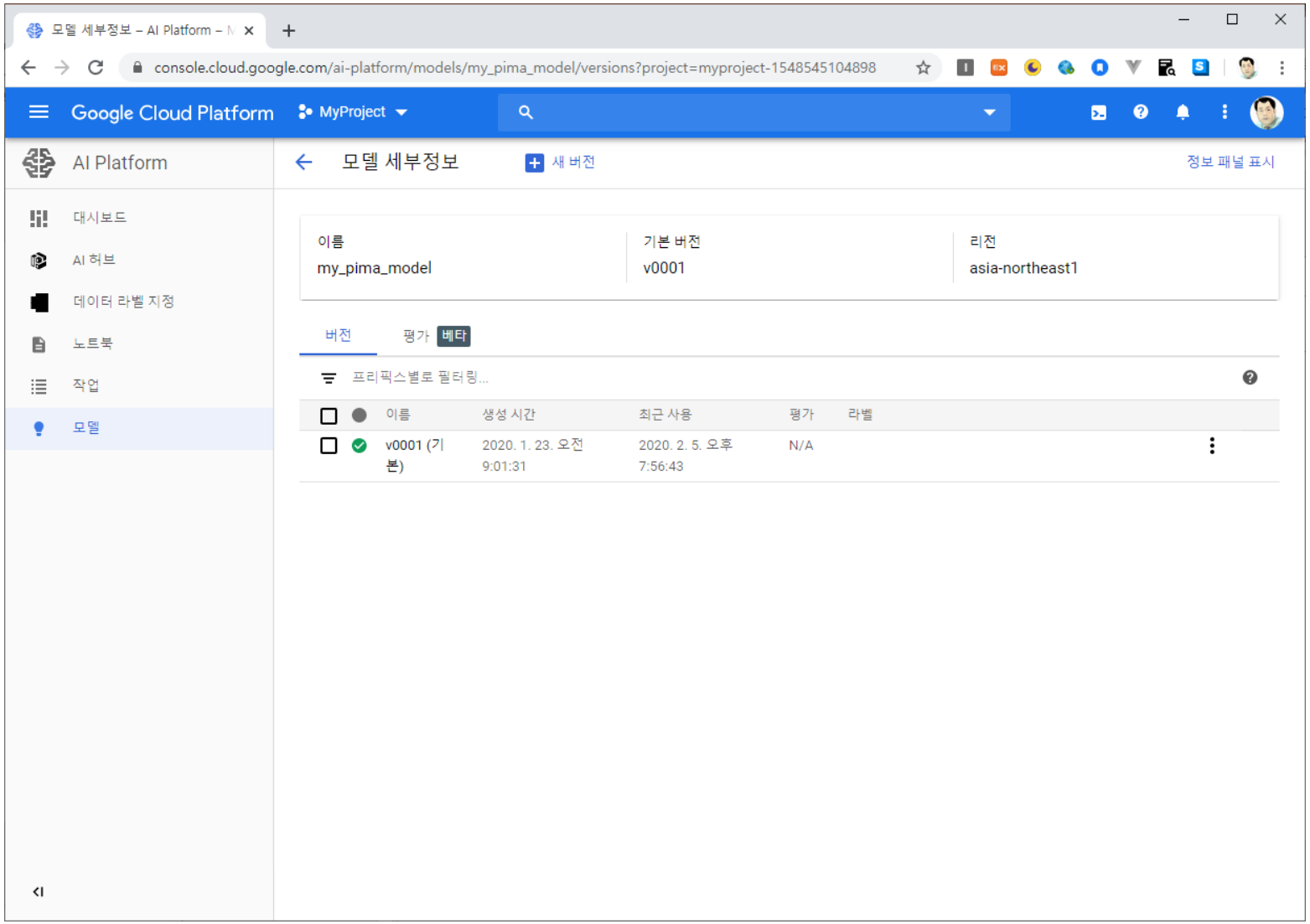
Google Cloud AI Platform에 Deployment



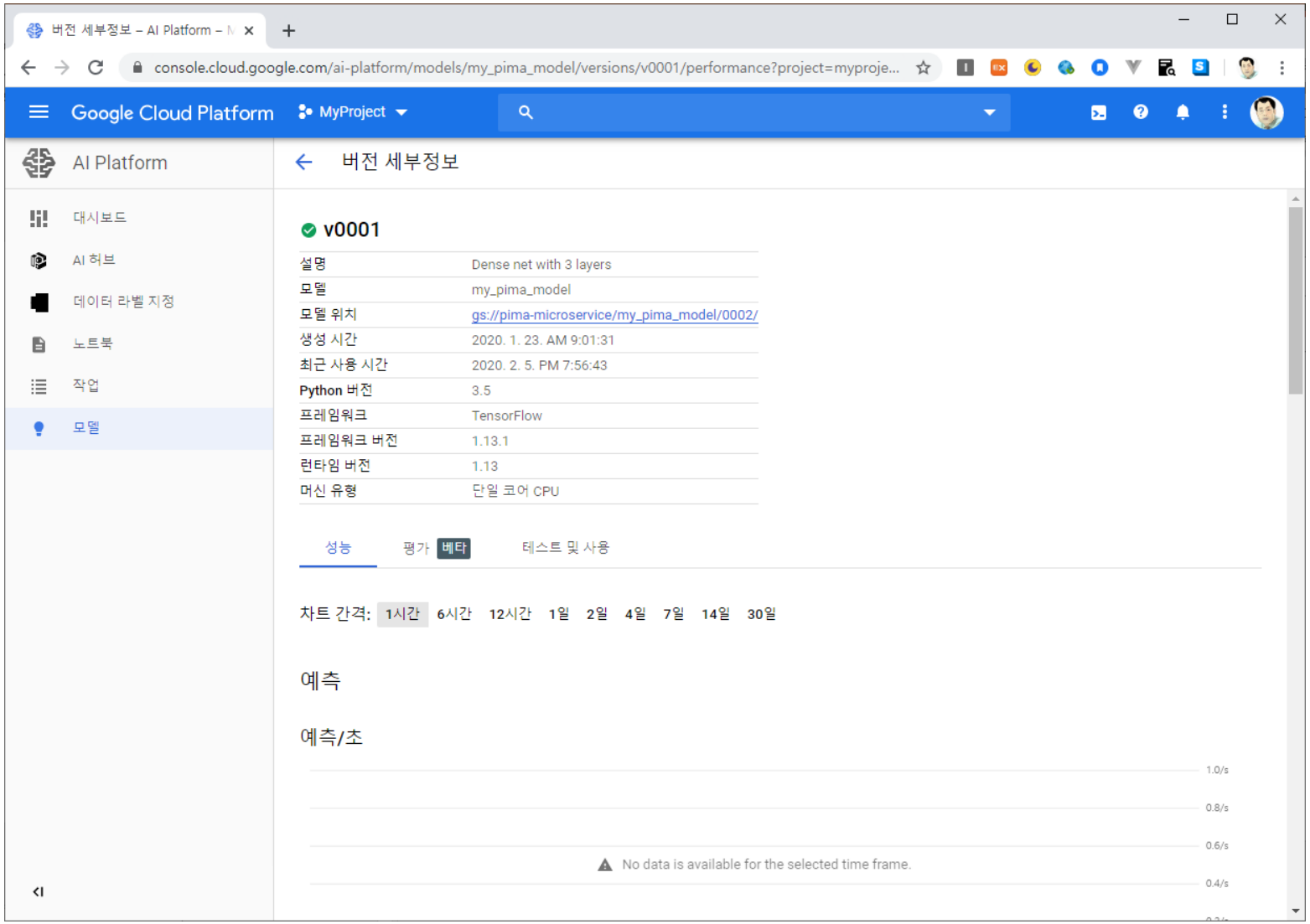
Google Cloud AI Platform에 Deployment



Google Cloud AI Platform에 Deployment



Google Cloud AI Platform에 Deployment



Cloud model client

```
print("And now let's use saved_model_cli to make predictions \n"
      "for the instances we just saved:")
print("let's start by creating the query.")

os.environ["GOOGLE_APPLICATION_CREDENTIALS"] = "myproject-                    6a9890f.json"

# set the variables for the gcp ai platform
project_id = "mypr                 04898"
model_id = "my_pima_model"
model_path = "projects/{}/models/{}".format(project_id, model_id)
model_path += "/versions/v0001/" # if you want to run a specific version
ml_resource = googleapiclient.discovery.build("ml", "v1").projects()

print("\nmodel_path: \n", model_path)
```

Cloud model client

```
# function use to request a prediction from the web api of the model
# and get a reponse of the predctions
def predict(X):
    input_data_json = {"signature_name": "serving_default",
                       "instances": X.tolist()}
    request = ml_resource.predict(name=model_path, body=input_data_json)
    response = request.execute()
    print("\nresponse: \n", response)

    if "error" in response:
        raise RuntimeError(response["error"])
    return np.array([pred['dense_2'] for pred in response["predictions"]])

print("\nX_test: \n", X_test)

# predict the results for the test set
print("\n\npredict(X_test[:3]): \n", predict(X_test[:3]))
```

Cloud model client

```
And now let's use saved_model_cli to make predictions
for the instances we just saved:
let's start by creating the query.

model_path:
 projects/myproject-1548545104898/models/my_pima_model/versions/v0001/

X_test:
 [[0.35294118 0.49246231 0.47540984 ... 0.50670641 0.15029889 0.36666667]
 [0.11764706 0.56281407 0.6147541  ... 0.53204173 0.02988898 0.        ]
 [0.11764706 0.54271357 0.52459016 ... 0.45901639 0.03415884 0.        ]
 ...
 [0.05882353 0.         0.60655738 ... 0.41281669 0.09436379 0.        ]
 [0.47058824 0.71859296 0.54098361 ... 0.52011923 0.02177626 0.33333333]
 [0.11764706 0.50753769 0.47540984 ... 0.36065574 0.22886422 0.03333333]]

response:
 {'predictions': [{'dense_2': [0.384774386882782]}, {'dense_2': [0.09595730155706406]}, {'dense_2': [0.0580522529780864 7]}]}


predict(X_test[:3]):
 [[0.38477439]
 [0.0959573 ]
 [0.05805225]]

Process finished with exit code 0
```

실습과제 15-3

본문에 나오는 전체 내용 PyCharm에서 실행하기

★ **반드시 결과 값 및 결과로 나온 모든 그래프에 대해 자세한 분석을 하시오.**

참고: 제 15강 실습과제 15-3 (cloud-model-client) How to build and deploy a diabetes diagnostic app - GCP.pdf