19953 Zhiyu Zhang

CS571: Week 10: Homework 1: Machine Learning on Kubernetes

This document covers the following steps to deploy a machine learning model on Kubernetes:

1. Train a ML model
2. Save and export the ML model
3. Create a Flask app including the UI layer
4. Set up Kubernetes and create a Dockerfile
5. Build the Docker image and run the Docker container locally
6. Stop the container
7. Deploy the app on Kubernetes

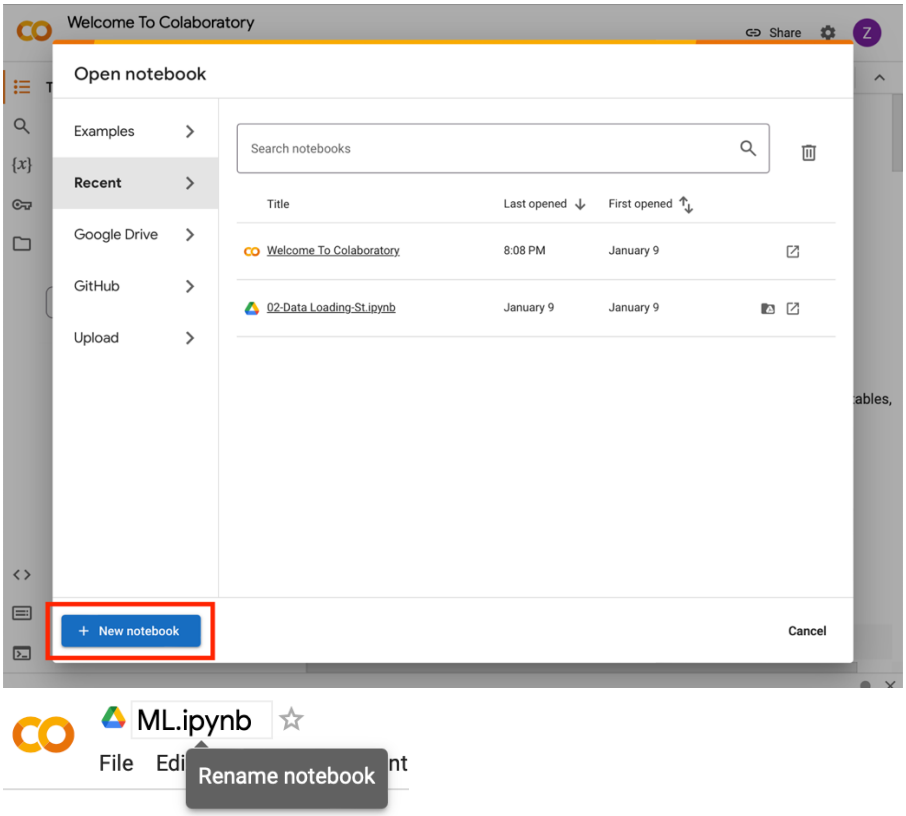
You can find all the files we need for this project here.

1. Train a ML model

1-1 Create a new notebook in Google Colab

Open Google Colab at https://colab.research.google.com.

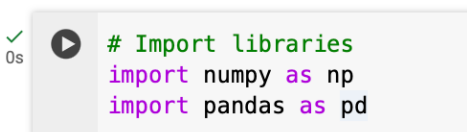
Create a new notebook and rename it as ML.ipynb:



1-2 Import libraries

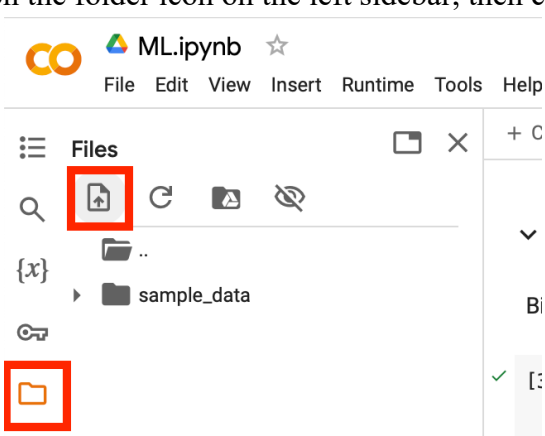
Import numpy and pandas to explore the dataset later:

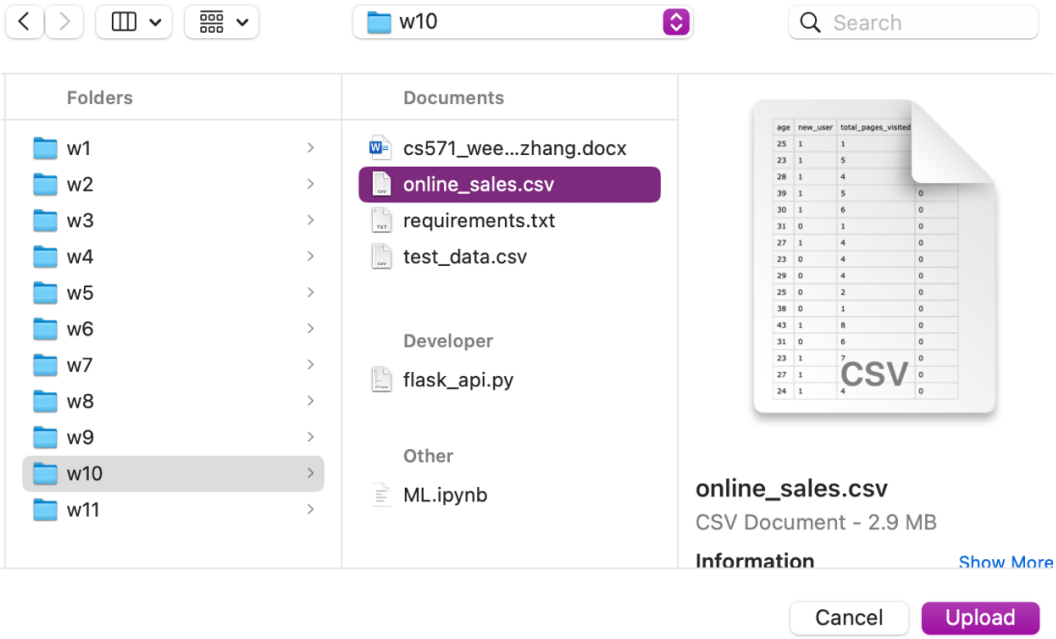
```
import numpy as np
import pandas as pd
```



1-3 Load dataset and study the data

Click on the folder icon on the left sidebar, then click upload icon to update “online_sales.csv” as our dataset:





Load the dataset and name it "df":

```
df = pd.read_csv('online_sales.csv')
```

```
[3] #read the data
    df = pd.read_csv('online_sales.csv')
```

See how the dataset looks and the class balance:

```
df.shape
df.head()
df.columns
df.converted.value_counts()
```

```
[4] df.shape
```

```
(316200, 4)
```

```
[5] df.head()
```

	age	new_user	total_pages_visited	converted
0	25	1	1	0
1	23	1	5	0
2	28	1	4	0
3	39	1	5	0
4	30	1	6	0

```
[6] df.columns
```

```
Index(['age', 'new_user', 'total_pages_visited', 'conver
```

```
[8] #target class frequency
    df.converted.value_counts()
```

```
0    306000
1     10200
Name: converted, dtype: int64
```

Check for missing values and get the general description of data:

```
df.info()
df.describe()
```

```
[9] df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 316200 entries, 0 to 316199
Data columns (total 4 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   age                  316200 non-null  int64
 1   new_user             316200 non-null  int64
 2   total_pages_visited  316200 non-null  int64
 3   converted            316200 non-null  int64
dtypes: int64(4)
memory usage: 9.6 MB
```

```
[10] df.describe()
```

	age	new_user	total_pages_visited	converted
count	316200.000000	316200.000000	316200.000000	316200.000000
mean	30.569858	0.685465	4.872966	0.032258
std	8.271802	0.464331	3.341104	0.176685
min	17.000000	0.000000	1.000000	0.000000
25%	24.000000	0.000000	2.000000	0.000000
50%	30.000000	1.000000	4.000000	0.000000
75%	36.000000	1.000000	7.000000	0.000000
max	123.000000	1.000000	29.000000	1.000000

1-4 Prepare data for modeling: identify input and output columns

Split data into input (features) and output (target) and display them:

```
input_columns = [column for column in df.columns if column != 'converted']
output_column = 'converted'
print (input_columns)
print (output_column)
```

```
[11] input_columns = [column for column in df.columns if column != 'converted']
     output_column = 'converted'
     print (input_columns)
     print (output_column)
```

```
['age', 'new_user', 'total_pages_visited']
converted
```

Convert the input and output into input features X and target variable y, and print the shape of them:

```
X = df.loc[:,input_columns].values
y = df.loc[:,output_column]
print (X.shape, y.shape)
```

```
[12] #input data
     X = df.loc[:,input_columns].values
     #output data
     y = df.loc[:,output_column]
     #shape of input and output dataset
     print (X.shape, y.shape)
```

```
(316200, 3) (316200,)
```

1-5 Split the dataset into train data and test data

Import train_test_split function and the Logistic Regression model from scikit-learn:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
```

```
[13] #import model specific libraries
     from sklearn.model_selection import train_test_split
     from sklearn.linear_model import LogisticRegression
```

Split the data into training and test data (70/30 ratio):

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3,
random_state=555, stratify=y)
```

```
[14] #Split the data into training and test data (70/30 ratio)
     X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3, random_state=555, stratify=y)
```

Validate the shape of train and test dataset:

```
print (X_train.shape)
print (y_train.shape)

print (X_test.shape)
print (y_test.shape)
```

```
[15] #validate the shape of train and test dataset
     print (X_train.shape)
     print (y_train.shape)

     print (X_test.shape)
     print (y_test.shape)

     (221340, 3)
     (221340,)
     (94860, 3)
     (94860,)
```

Check on number of positive classes in train and test dataset:

```
print(np.sum(y_train))
print(np.sum(y_test))
```

```
[16] #check on number of positive classes in train and test data set
     print(np.sum(y_train))
     print(np.sum(y_test))

     7140
     3060
```

1-6 Train the logistic model

Fit the logisitc regression model on training dataset:

```
logreg = LogisticRegression(class_weight='balanced').fit(X_train,y_train)
```

```
[17] #fit the logisitc regression model on training dataset
     logreg = LogisticRegression(class_weight='balanced').fit(X_train,y_train)
```

Check the model performance on train dataset:

```
logreg.score(X_train, y_train)
```

```
[18] logreg.score(X_train, y_train)
```

```
0.9370470768952742
```

Validate the model performance on unseen data:

```
logreg.score(X_test, y_test)
```

```
[19] #validate the model performance on unseen data
     logreg.score(X_test, y_test)
```

```
0.9369175627240144
```

Make predictions on unseen data:

```
predictions=logreg.predict(X_test)
```

```
[20] #make predictions on unseen data
     predictions=logreg.predict(X_test)
```

1-7 Get the model and the result report

Use confusion matrix to assess the model and generate a report:

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
print(classification_report(y_test, predictions,target_names=["Non Converted",
"Converted"]))
```

```
[21] from sklearn.metrics import confusion_matrix
     from sklearn.metrics import classification_report

     print(classification_report(y_test, predictions,target_names=["Non Converted", "Converted"]))
```

```
              precision    recall  f1-score   support

Non Converted       1.00      0.94      0.97     91800
    Converted       0.33      0.92      0.49      3060

     accuracy                           0.94     94860
    macro avg       0.66      0.93      0.73     94860
 weighted avg       0.98      0.94      0.95     94860
```

Display the model:

```
logreg
```

```
logreg
```

```
LogisticRegression
LogisticRegression(class_weight='balanced')
```

2. Save and export the ML model

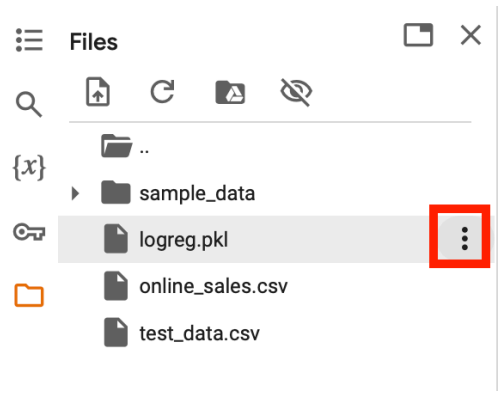
2-1 Export the model by serializing the model with Pickle

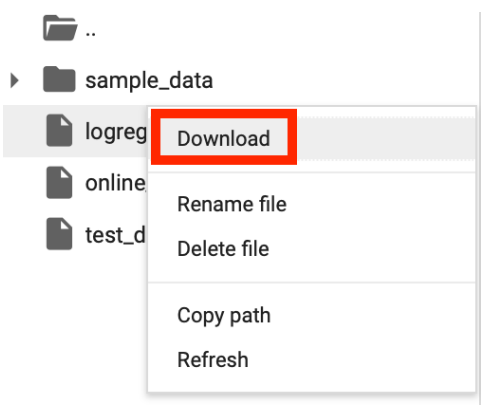
Create a Pickle file using serialization. “logreg” is the Machine Learning Model object; “logreg.pkl” is a file storing the serialized logreg:

```
import pickle
pickle_out = open("logreg.pkl","wb")
pickle.dump(logreg, pickle_out)
pickle_out.close()
```

```
[23] ### Create a Pickle file using serialization
     import pickle

     pickle_out = open("logreg.pkl","wb")
     pickle.dump(logreg, pickle_out)
     pickle_out.close()
```

Remember to download the model from Colab to a local directory so we can use it in the future:





2-2 Load the model for prediction

```
pickle_in = open("logreg.pkl","rb")
model = pickle.load(pickle_in)
```

```
[24] pickle_in = open("logreg.pkl","rb")
     model = pickle.load(pickle_in)
```

```
[25] model
```

```
LogisticRegression
LogisticRegression(class_weight='balanced')
```

2-3 Single customer prediction

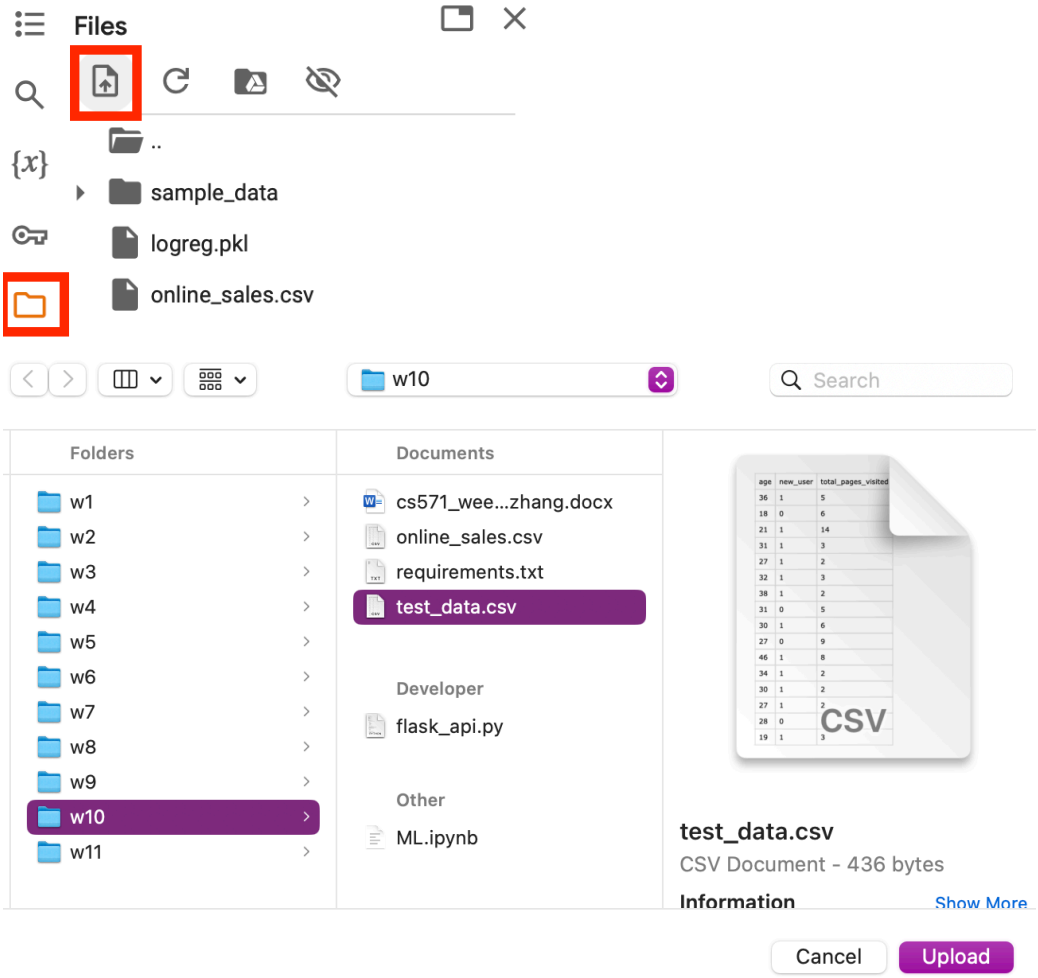
```
model.predict([[32,1,1]])[0]
```

```
[26] #predict using the model on customer input
     model.predict([[32,1,1]])[0]
```

```
0
```

2-4 Group customer prediction

Upload “test_data.csv” for prediction:



Load the dataset and name it “df_test”:

```
df_test = pd.read_csv('test_data.csv')
```

Make predictions and print them out:

```
predictions = model.predict(df_test)
print(list(predictions))
```

```
#Group prediction (multiple customers)
df_test = pd.read_csv('test_data.csv')
predictions = model.predict(df_test)

print(list(predictions))
```

```
[0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:432: UserWarning: X has feature names, but LogisticRegression was fitted without feature names
  warnings.warn(
```

3. Create a Flask app including the UI layer

3-1 Install required modules in python environment

Open terminal and run the following commands:

```
% pip install Flask
% pip install numpy
% pip install pandas
% pip install flasgger
```

```
(base) zhangzhiyu@shengchligongju ~ % pip install Flask
pip install numpy
pip install pandas
pip install flasgger

Requirement already satisfied: Flask in ./anaconda3/lib/python3.11/site-packages (2.2.2)
Requirement already satisfied: Werkzeug>=2.2.2 in ./anaconda3/lib/python3.11/site-packages (from Flask) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in ./anaconda3/lib/python3.11/site-packages (from Flask) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in ./anaconda3/lib/python3.11/site-packages (from Flask) (2.0.1)
Requirement already satisfied: click>=8.0 in ./anaconda3/lib/python3.11/site-packages (from Flask) (8.0.4)
Requirement already satisfied: MarkupSafe>=2.0 in ./anaconda3/lib/python3.11/site-packages (from Jinja2>=3.0->Flask) (2.1.1)
Requirement already satisfied: numpy in ./anaconda3/lib/python3.11/site-packages (1.24.3)
Requirement already satisfied: pandas in ./anaconda3/lib/python3.11/site-packages (2.0.3)
Requirement already satisfied: python-dateutil>=2.8.2 in ./anaconda3/lib/python3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in ./anaconda3/lib/python3.11/site-packages (from pandas) (2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in ./anaconda3/lib/python3.11/site-packages (from pandas) (2023.3)
Requirement already satisfied: numpy>=1.21.0 in ./anaconda3/lib/python3.11/site-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in ./anaconda3/lib/python3.11/site-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Collecting flasgger
8  Downloading flasgger-0.9.7.1.tar.gz (4.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.0/4.0 MB 2.2 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: Flask>=0.10 in ./anaconda3/lib/python3.11/site-packages (from flasgger) (2.2.2)
Requirement already satisfied: PyYAML>=3.0 in ./anaconda3/lib/python3.11/site-packages (from flasgger) (6.0)
Requirement already satisfied: jsonschema>=3.0.1 in ./anaconda3/lib/python3.11/site-packages (from flasgger) (4.17.3)
Requirement already satisfied: mistune in ./anaconda3/lib/python3.11/site-packages (from flasgger) (0.8.4)
Requirement already satisfied: six>=1.10.0 in ./anaconda3/lib/python3.11/site-packages (from flasgger) (1.16.0)
Requirement already satisfied: packaging in ./anaconda3/lib/python3.11/site-packages (from flasgger) (23.1)
Requirement already satisfied: Werkzeug>=2.2.2 in ./anaconda3/lib/python3.11/site-packages (from Flask>=0.10->flasgger) (2.2.3)
Requirement already satisfied: Jinja2>=3.0 in ./anaconda3/lib/python3.11/site-packages (from Flask>=0.10->flasgger) (3.1.2)
Requirement already satisfied: itsdangerous>=2.0 in ./anaconda3/lib/python3.11/site-packages (from Flask>=0.10->flasgger) (2.0.1)
Requirement already satisfied: click>=8.0 in ./anaconda3/lib/python3.11/site-packages (from Flask>=0.10->flasgger) (8.0.4)
Requirement already satisfied: attrs>=17.4.0 in ./anaconda3/lib/python3.11/site-packages (from jsonschema>=3.0.1->flasgger) (22.1.0
)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in ./anaconda3/lib/python3.11/site-packages (from json
schema>=3.0.1->flasgger) (0.18.0)
Requirement already satisfied: MarkupSafe>=2.0 in ./anaconda3/lib/python3.11/site-packages (from Jinja2>=3.0->Flask>=0.10->flasgger
) (2.1.1)
Building wheels for collected packages: flasgger
  Building wheel for flasgger (setup.py) ... done
  Created wheel for flasgger: filename=flasgger-0.9.7.1-py2.py3-none-any.whl size=3865185 sha256=a1a7299ad218bc36efb13833f35a1be877
4e2b38ec036bcac78874695bd85864
  Stored in directory: /Users/zhangzhiyu/Library/Caches/pip/wheels/da/2d/f6/c125d4b8701ac49fdb2abbfb978036c46d5ad64d2ffaa2ac34
Successfully built flasgger
Installing collected packages: flasgger
Successfully installed flasgger-0.9.7.1
```

3-2 Create a python file and import the required libraries

```
from flask import Flask, request
import numpy as np
import pickle
import pandas as pd
import flasgger
from flasgger import Swagger
```

3-3 Create the Flask app and wrap it around Swagger

```
app=Flask(__name__)
Swagger(app)
```

3-4 Load machine learning model using Pickle's deserialization feature

```
pickle_in = open("logreg.pkl","rb")
model=pickle.load(pickle_in)
```

3-5 Define the function prediction_class() for parameter-based prediction via a GET request

```
@app.route('/predict',methods=["GET"])
def predict_class():

    """Predict if Customer would buy the product or not .
    ---
    parameters:
      - name: age
        in: query
        type: number
        required: true
      - name: new_user
        in: query
        type: number
        required: true
      - name: total_pages_visited
        in: query
        type: number
        required: true

    responses:
        500:
            description: Prediction

    """
    age=int(request.args.get("age"))
    new_user=int(request.args.get("new_user"))
    total_pages_visited=int(request.args.get("total_pages_visited"))
    prediction=model.predict([[age,new_user,total_pages_visited]])
    print(prediction[0])
    return "Model prediction is"+str(prediction)
```

3-6 Define the function prediction_test_file() for file-based prediction via a POST request

```
@app.route('/predict_file',methods=["POST"])
def prediction_test_file():
    """Prediction on multiple input test file .
    ---
    parameters:
      - name: file
        in: formData
        type: file
        required: true

    responses:
        500:
            description: Test file Prediction

    """
    df_test=pd.read_csv(request.files.get("file"))
    prediction=model.predict(df_test)

    return str(list(prediction))
```

3-7 Define the main function

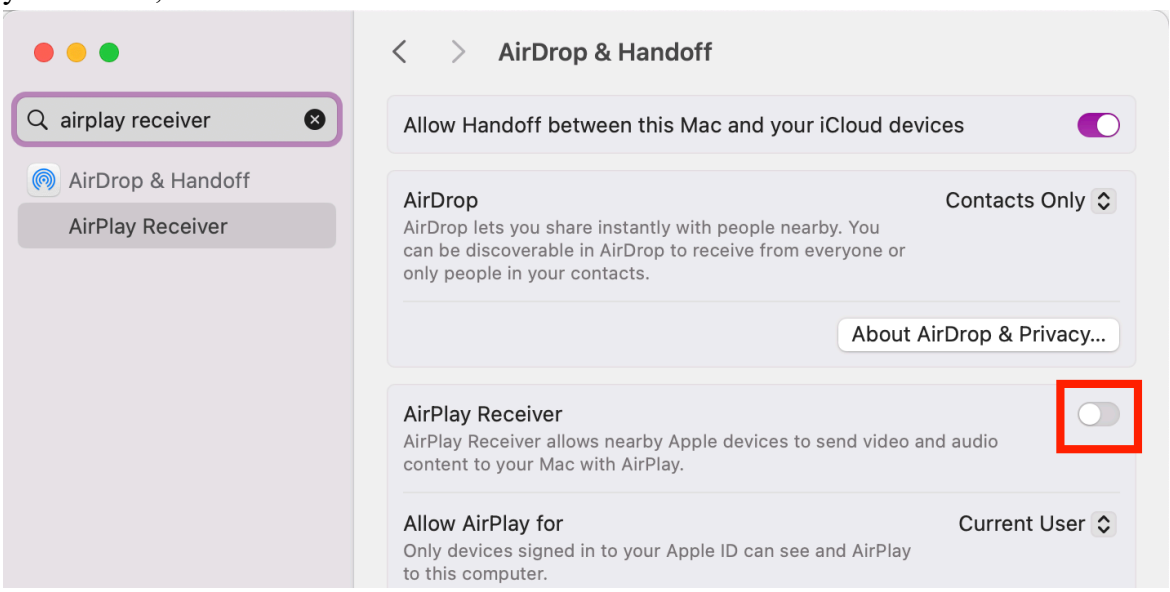
```
if __name__=='__main__':
    app.run(debug=True,host='0.0.0.0')
```

3-8 Possible error

When you try to run this program locally, if you're using a MacOS machine, you might encounter this "Address already in use" error:

```
  warnings.warn(
 * Serving Flask app 'flask_api'
 * Debug mode: on
Address already in use
Port 5000 is in use by another program. Either identify and stop that program, or star
t the server with a different port.
On macOS, try disabling the 'AirPlay Receiver' service from System Preferences -> Shar
ing.
```

This is because MacOS uses port 5000 as airplay receiver. To solve this, open System Settings and search for "Airplay Receiver", then turn off this function:



You can enable it again once you make sure the program is running normally. When it's running normally, you'll see something like this:

```
  warnings.warn(
 * Serving Flask app 'flask_api'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a
 production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://192.168.1.197:5000
Press CTRL+C to quit
 * Restarting with watchdog (fsevents)
/Users/zhangzhiyu/anaconda3/lib/python3.11/site-packages/sklearn/base.py:347: Inconsis
tentVersionWarning: Trying to unpickle estimator LogisticRegression from version 1.2.2
 when using version 1.3.0. This might lead to breaking code or invalid results. Use at
 your own risk. For more info please refer to:
https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limita
tions
  warnings.warn(
 * Debugger is active!
 * Debugger PIN: 616-197-969
```

4. Set up Kubernetes and create a Dockerfile

4-1 Create a requirements.txt

With the following code, create requiments.txt that contains the required libraries and versions:

```
Flask==1.1.1
gunicorn==19.9.0
itsdangerous==1.1.0
Jinja2==2.10.1
MarkupSafe==1.1.1
Werkzeug==0.15.5
numpy>=1.9.2
scipy>=0.15.1
scikit-learn==0.22.1
matplotlib>=1.4.3
pandas>=0.19
flasgger==0.9.4
```

4-2 In GCP terminal, create a Kubernetes cluster with three nodes

```
$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1
```

```
zzhang42305@cloudshell:~ (cs571-a)$ gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=
us-west1
Default change: VPC-native is the default mode during cluster creation for versions greater than 1.21.0-gke.1500. To creat
e advanced routes based clusters, please pass the `--no-enable-ip-alias` flag
Note: Your Pod address range (`--cluster-ipv4-cidr`) can accommodate at most 1008 node(s).
Creating cluster kubia in us-west1... Cluster is being health-checked (master is healthy)...done.
Created [https://container.googleapis.com/v1/projects/cs571-a/zones/us-west1/clusters/kubia].
To inspect the contents of your cluster, go to: https://console.cloud.google.com/kubernetes/workload_/gcloud/us-west1/kubi
a?project=cs571-a
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.203.179.66
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```

4-3 Make sure cluster is running

```
$ gcloud container clusters list
```

```
zzhang42305@cloudshell:~ (cs571-a)$ gcloud container clusters list
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.203.179.66
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
```

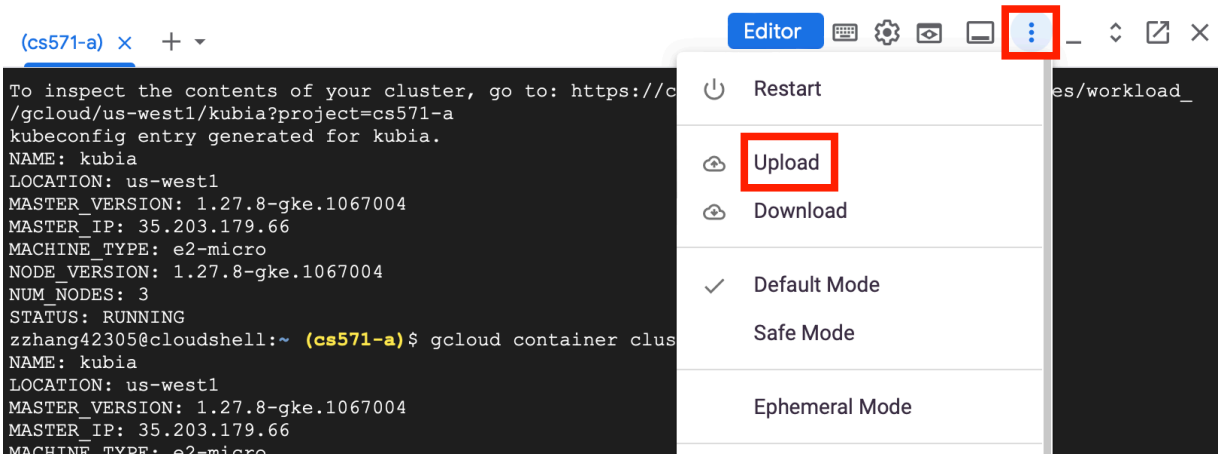
4-4 Make sure all nodes are running

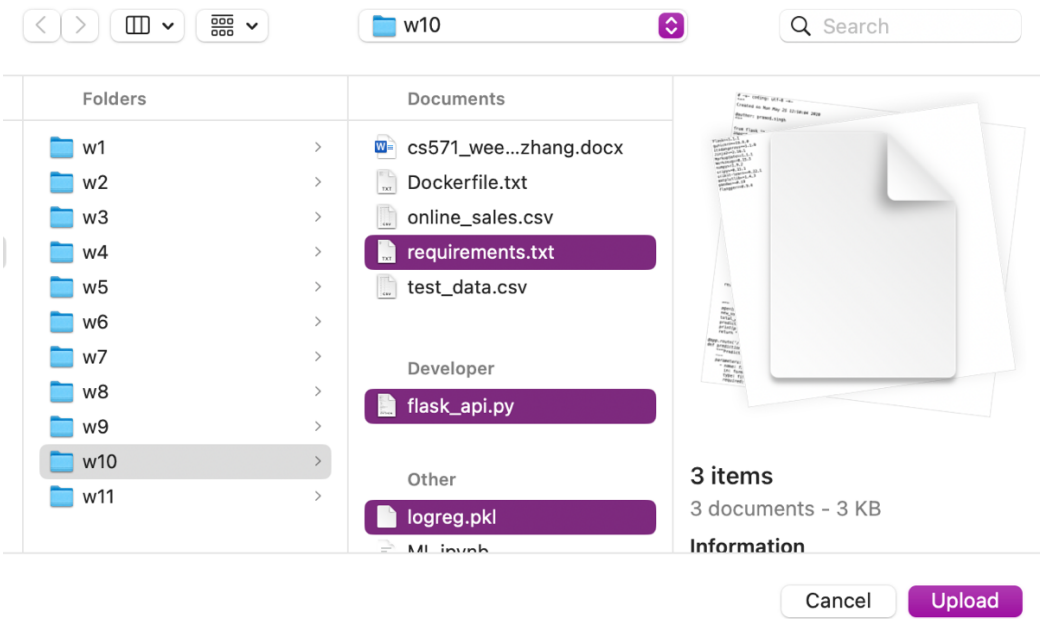
```
$ kubectl get nodes
```

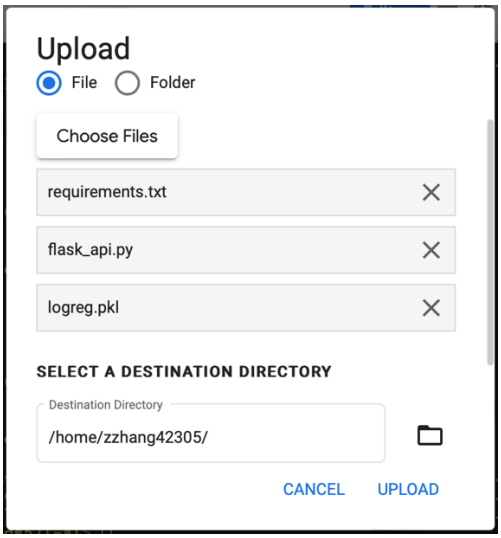
```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get nodes
NAME                                          STATUS   ROLES    AGE     VERSION
gke-kubia-default-pool-11d4ce1f-83j2          Ready    <none>   2m27s   v1.27.8-gke.1067004
gke-kubia-default-pool-49dc988d-r9hd          Ready    <none>   2m28s   v1.27.8-gke.1067004
gke-kubia-default-pool-c2b72ed3-8qb9          Ready    <none>   2m26s   v1.27.8-gke.1067004
```

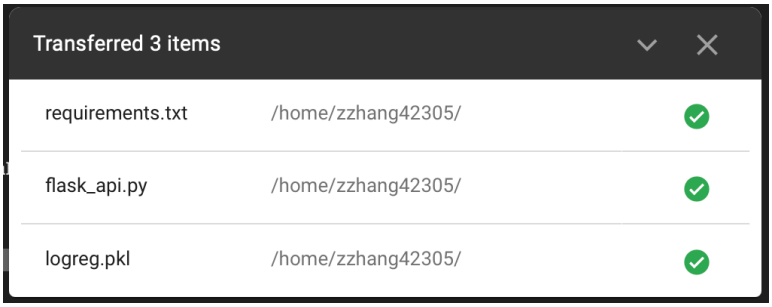
4-5 Upload files

Click the 3 dots on the top right corner, then upload the 3 files (the ML model, the Flask app, and the requirements for Dockerfile) we need as shown below:


(cs571-a)
Editor
To inspect the contents of your cluster, go to: https://c
/gcloud/us-west1/kubia?project=cs571-a
kubeconfig entry generated for kubia.
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.203.179.66
MACHINE_TYPE: e2-micro
NODE_VERSION: 1.27.8-gke.1067004
NUM_NODES: 3
STATUS: RUNNING
zzhang42305@cloudshell:~ (cs571-a)$ gcloud container clus
NAME: kubia
LOCATION: us-west1
MASTER_VERSION: 1.27.8-gke.1067004
MASTER_IP: 35.203.179.66
es/workload_
Restart
Upload
Download
Default Mode
Safe Mode
Ephemeral Mode


w10
Search
Folders
w1
w2
w3
w4
w5
w6
w7
w8
w9
w10
w11
Documents
cs571_wee...zhang.docx
Dockerfile.txt
online_sales.csv
requirements.txt
test_data.csv
Developer
flask_api.py
Other
logreg.pkl
3 items
3 documents - 3 KB
Information
Cancel
Upload


Upload
File
Folder
Choose Files
requirements.txt
flask_api.py
logreg.pkl
SELECT A DESTINATION DIRECTORY
Destination Directory
/home/zzhang42305/
CANCEL
UPLOAD

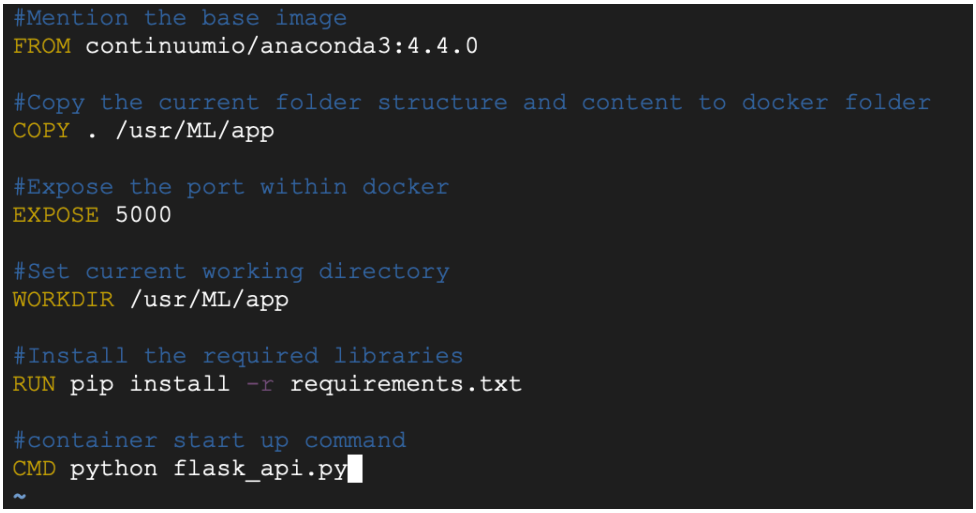

Transferred 3 items
requirements.txt
/home/zzhang42305/
flask_api.py
/home/zzhang42305/
logreg.pkl
/home/zzhang42305/

4-6 Create Dockerfile

```
$ vim Dockerfile
```

with the following code:

```
#Mention the base image
FROM continuumio/anaconda3:4.4.0

#Copy the current folder structure and content to docker folder
COPY . /usr/ML/app

#Expose the port within docker
EXPOSE 5000

#Set current working directory
WORKDIR /usr/ML/app

#Install the required libraries
RUN pip install -r requirements.txt

#container start up command
CMD python flask_api.py
```



5. Build the Docker image and run the Docker container locally

5-1 Build the docker image

```
$ docker build -t ml_app_docker .
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker build -t ml_app_docker .
[+] Building 14.5s (9/9) FINISHED                                                    docker:default
 => [internal] load build definition from Dockerfile                                          0.0s
 => => transferring dockerfile: 400B                                                          0.0s
 => [internal] load metadata for docker.io/continuumio/anaconda3:4.4.0                        0.1s
 => [internal] load .dockerignore                                                             0.0s
 => => transferring context: 2B                                                               0.0s
 => [internal] load build context                                                             0.1s
 => => transferring context: 153.38kB                                                         0.1s
 => CACHED [1/4] FROM docker.io/continuumio/anaconda3:4.4.0@sha256:c6bb52bffe028b4b436b085afa4  0.0s
 => [2/4] COPY . /usr/ML/app                                                                  1.2s
 => [3/4] WORKDIR /usr/ML/app                                                                 0.0s
 => [4/4] RUN pip install -r requirements.txt                                                12.3s
 => exporting to image                                                                        0.6s
 => => exporting layers                                                                       0.6s
 => => writing image sha256:e78adf25aed616f4c54df7704d1250504a09aa3300322ca4faf00c0226a415b8   0.0s
 => => naming to docker.io/library/ml_app_docker                                              0.0s
```

5-2 Run the image in a local container

```
$ docker container run -p 5000:5000 ml_app_docker
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker container run -p 5000:5000 ml_app_docker
 * Serving Flask app "flask_api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator
 LogisticRegression from version 1.2.2 when using version 0.22.1. This might lead to breaking code or
 invalid results. Use at your own risk.
  UserWarning)
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
/opt/conda/lib/python3.6/site-packages/sklearn/base.py:318: UserWarning: Trying to unpickle estimator
 LogisticRegression from version 1.2.2 when using version 0.22.1. This might lead to breaking code or
 invalid results. Use at your own risk.
  UserWarning)
 * Debugger is active!
 * Debugger PIN: 113-556-481
```

5-3 Access the app locally using curl

Open another terminal and type:

```
$ curl localhost:5000/apidocs/
```

Since GCP terminal doesn't have a graphic interface, and right now we can only access the app within GCP terminal, you can only see the HTML content of the Swagger UI page, which starts with something like this:

```
zzhang42305@cloudshell:~ (cs571-a)$ curl localhost:5000/apidocs/
<!-- HTML for static distribution bundle build -->
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
<title>Flasgger</title>
<link href="https://fonts.googleapis.com/css?family=Open+Sans:400,700|Source+Code+Pro:300,600|Titillium+Web
:400,600,700" rel="stylesheet">
<link rel="stylesheet" type="text/css" href="/flasgger_static/swagger-ui.css">
<!-- Customize the app.config['SWAGGER']['favicon'] -->
<link rel="icon" type="image/png" href="/flasgger_static/favicon-32x32.png" sizes="64x64 32x32 16x16" />
<style>
    html
    {
    box-sizing: border-box;
    overflow: -moz-scrollbars-vertical;
    overflow-y: scroll;
    }

    *,
    *:before,
    *:after
    {
    box-sizing: inherit;
    }

    body
```

6. Stop the container

6-1 List running containers

```
$ docker ps
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                    NAMES
4e36d43e8ccf   ml_app_docker  "/usr/bin/tini -- /b…"   36 minutes ago   Up 36 minutes   0.0.0.0:5000->5000/tcp   goofy_herschel
```

6-2 select the running container ID to stop it

```
$ docker kill <Container_ID>
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker kill 4e36d43e8ccf
4e36d43e8ccf
```

7. Deploy the app on Kubernetes

Now let's deploy this image on Kubernetes so we can access the Flask App using a web browser.

7-1 Log in Docker hub

```
$ docker login -u username
```

then enter your password at the following prompt:

```
zzhang42305@cloudshell:~ (cs571-a)$ docker login -u zoeyzhiyu
Password:
WARNING! Your password will be stored unencrypted in /home/zzhang42305/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
```

7-2 Tag the docker image

```
$ docker tag ml_app_docker zoeyzhiyu/ml_app_docker:latest
$ docker push zoeyzhiyu/ml_app_docker:latest
```

```
zzhang42305@cloudshell:~ (cs571-a)$ docker tag ml_app_docker zoeyzhiyu/ml_app_docker:latest
zzhang42305@cloudshell:~ (cs571-a)$ docker push zoeyzhiyu/ml_app_docker:latest
The push refers to repository [docker.io/zoeyzhiyu/ml_app_docker]
077f8fe4e6b8: Pushed
5f70bf18a086: Pushed
5f13f5f20223: Pushed
ca173dc10e31: Mounted from continuumio/anaconda3
54e10c08a841: Mounted from continuumio/anaconda3
1f09b1beaa90: Mounted from continuumio/anaconda3
9e63c5bce458: Mounted from continuumio/anaconda3
latest: digest: sha256:f11281d91030ac41d0ce6a9bbd53b67900850c774bc9f70ecb82755b04893963 size: 1794
```

7-3 Deploy the image

Create a ml-app-deployment.yaml

```
$ vim ml-app-deployment.yaml
```

with the following code:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-app-deployment
spec:
  replicas: 3  # You can adjust the number of replicas
  selector:
    matchLabels:
      app: ml-app
  template:
    metadata:
      labels:
        app: ml-app
    spec:
      containers:
      - name: ml-app
        image: zoeyzhiyu/ml_app_docker:latest
        ports:
        - containerPort: 5000
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-app-deployment
spec:
  replicas: 3  # You can adjust the number of replicas
  selector:
    matchLabels:
      app: ml-app
  template:
    metadata:
      labels:
        app: ml-app
    spec:
      containers:
      - name: ml-app
        image: zoeyzhiyu/ml_app_docker:latest
        ports:
        - containerPort: 5000
```

Then create the deployment:

```
$ kubectl apply -f ml-app-deployment.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim ml-app-deployment.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f ml-app-deployment.yaml
deployment.apps/ml-app-deployment created
```

7-4 Use LoadBalancer to expose the service

Create a ml-app-service.yaml

```
$ vim ml-app-service.yaml
```

with the following code:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: ml-app-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 5000
  selector:
    app: ml-app
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: ml-app-service
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 5000
  selector:
    app: ml-app
```

Then create the service:

```
$ kubectl apply -f ml-app-service.yaml
```

```
zzhang42305@cloudshell:~ (cs571-a)$ vim ml-app-service.yaml
zzhang42305@cloudshell:~ (cs571-a)$ kubectl apply -f ml-app-service.yaml
service/ml-app-service created
```

7-5 Make sure all pods are ready and running

```
$ kubectl get pods
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get pods
NAME                                  READY   STATUS    RESTARTS   AGE
ml-app-deployment-5f5fb889c9-7c6n9    1/1     Running   0          15m
ml-app-deployment-5f5fb889c9-cp7bc    1/1     Running   0          15m
ml-app-deployment-5f5fb889c9-ktwxv    1/1     Running   0          15m
```

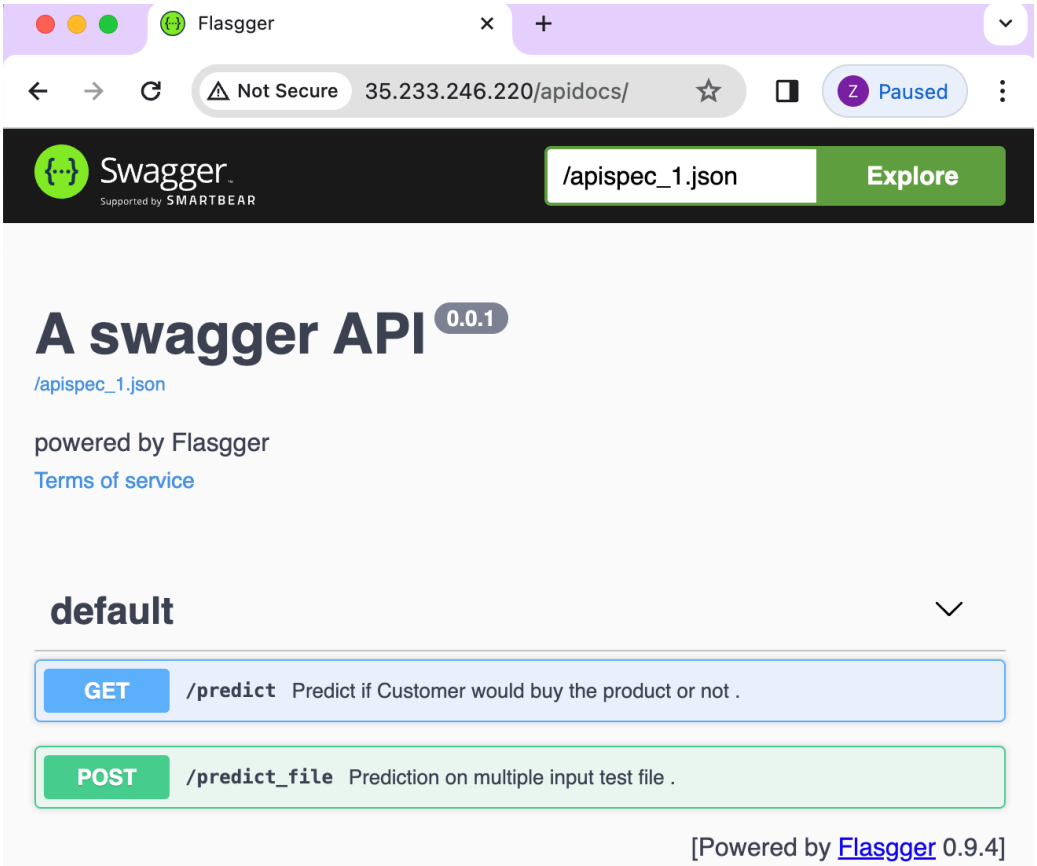
7-6 Get the external IP address of the service

```
$ kubectl get services
```

```
zzhang42305@cloudshell:~ (cs571-a)$ kubectl get services
NAME             TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
kubernetes       ClusterIP      10.121.128.1     <none>           443/TCP        75m
ml-app-service   LoadBalancer   10.121.143.172   35.233.246.220   80:32206/TCP   55s
```

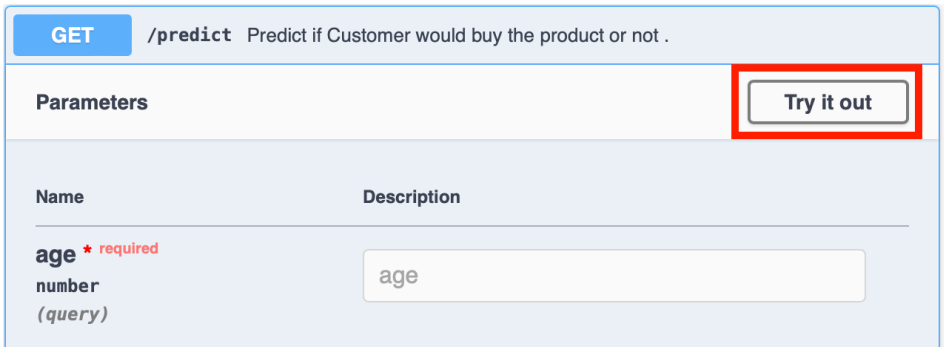
7-7 Access the app with the external IP address

Open a browser and go to http://EXTERNAL_IP/apidocs/

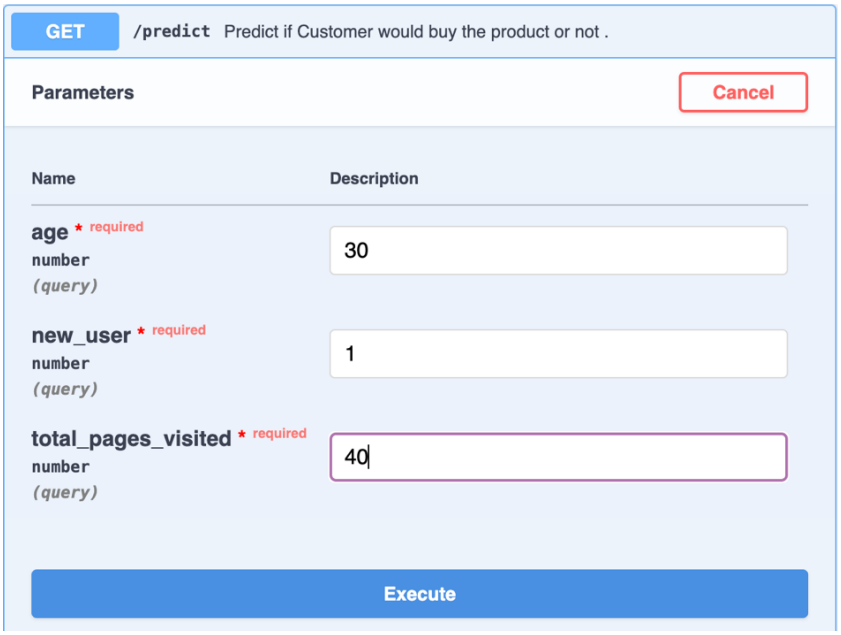


7-8 Use GET tab to make single customer prediction

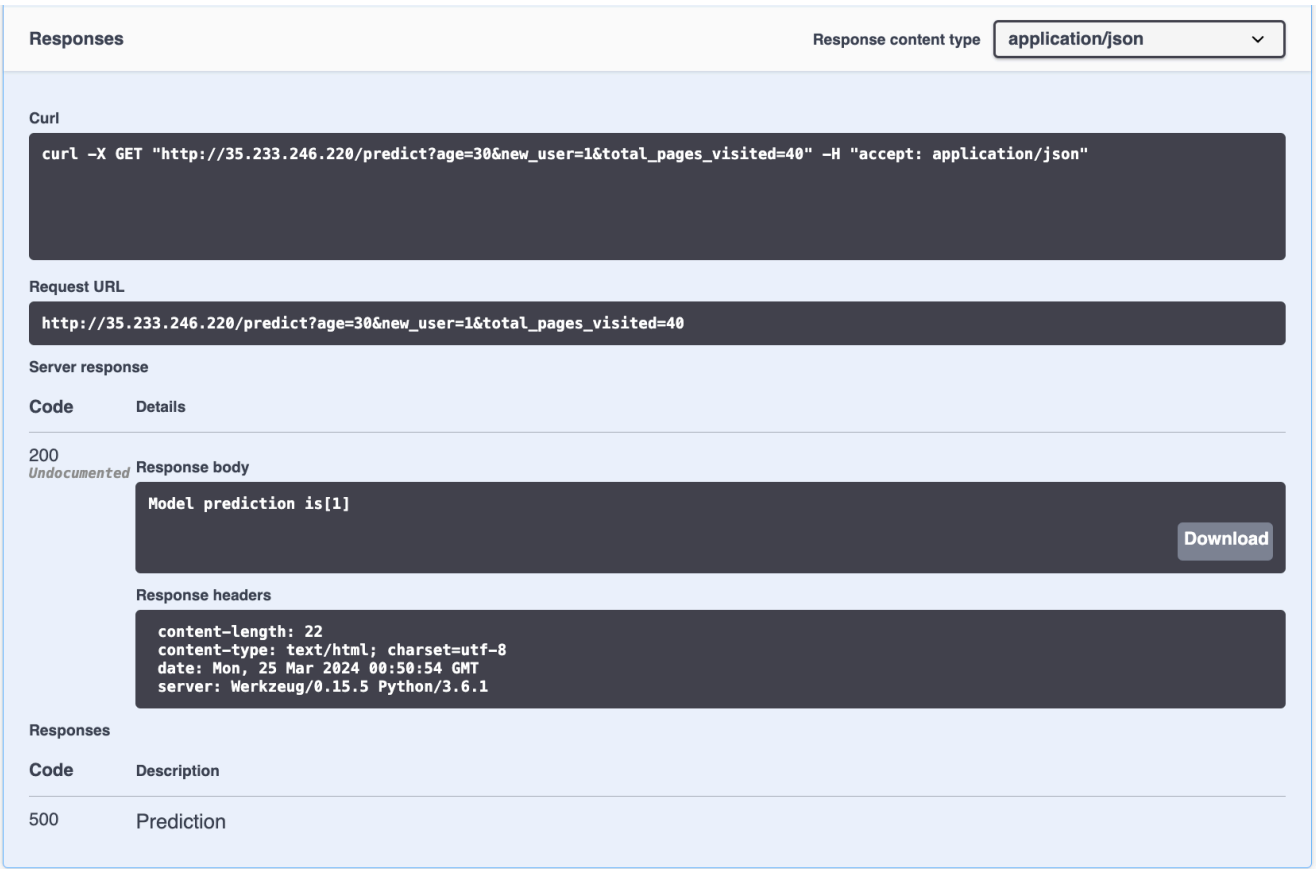
Click on the GET tab, then click “Try it out”:



Enter the required data, then click “Execute”:

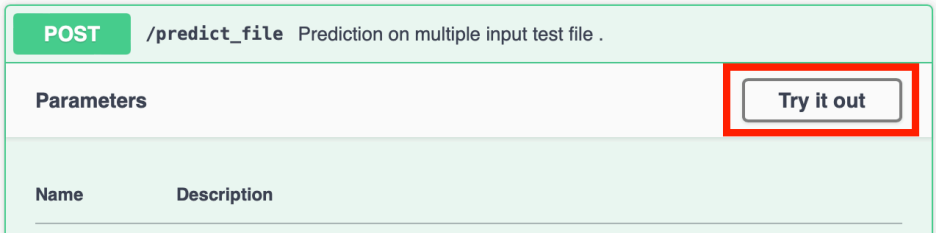


The prediction is displayed under “Responses”:

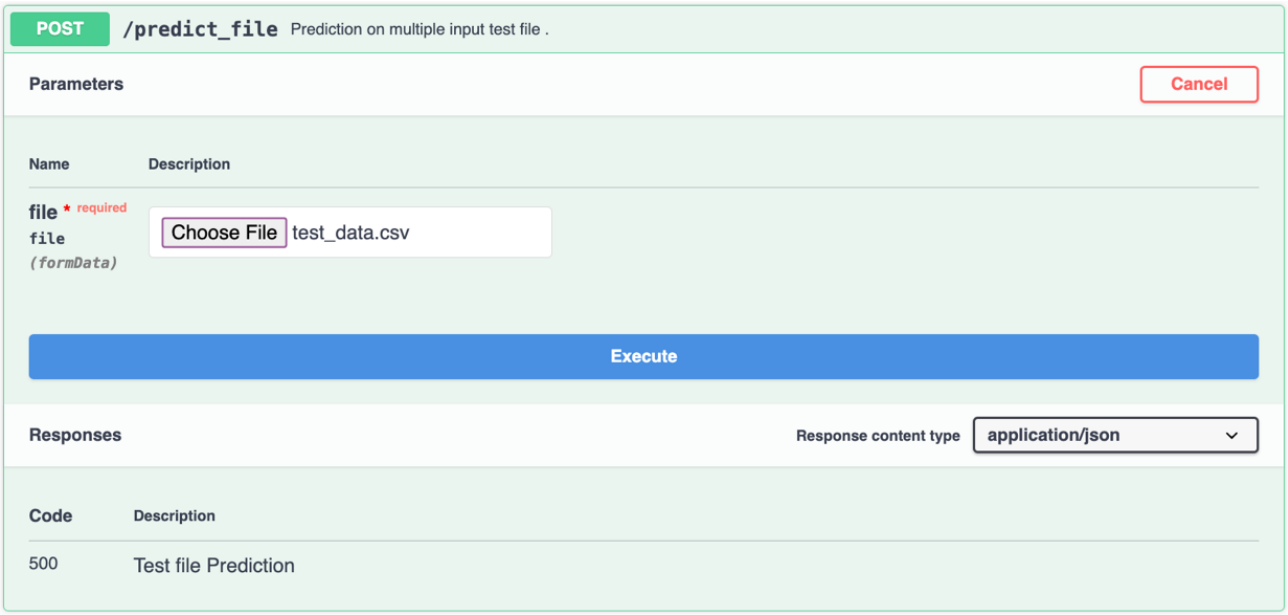
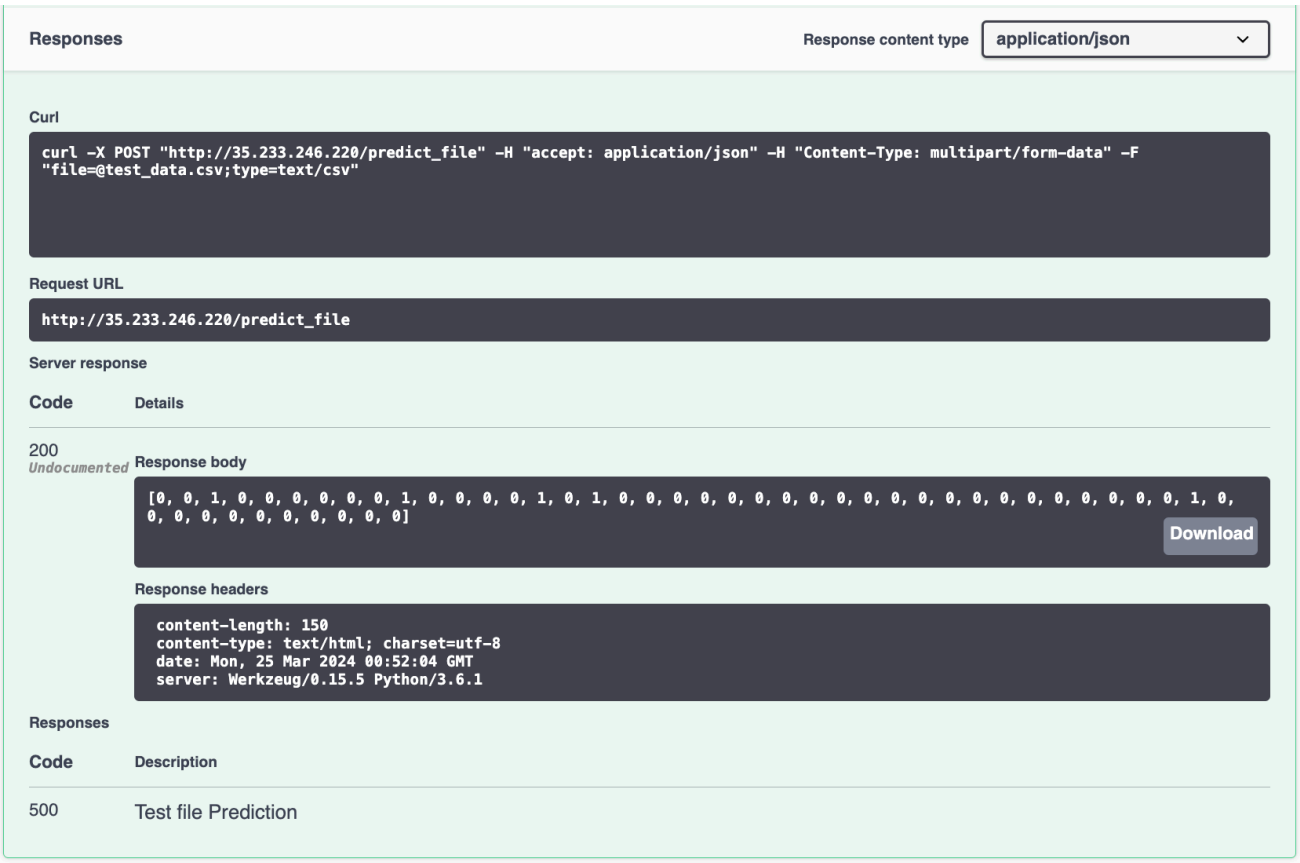


7-9 Use POST tab to make group customer prediction

Click on the POST tab, then click “Try it out”:



Upload test_data.csv as our dataset for group customer prediction, then click “Execute”:



The prediction is displayed under “Responses”: