Integrate Flask with Scoring End Point

app.py

```
# import the necessary packages
import pandas as pd
import numpy as np
import pickle
import os
from flask import Flask,request, render_template
app=Flask(_name_,template_folder="templates")
@app.route('/', methods=['GET'])
def index():
    return  render_template('home.html')
@app.route('/home', methods=['GET'])
def about():
    return render_template('home.html')
@app.route('/pred',methods=['GET'])
def page():
    return render_template('upload.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    print("[INFO] loading model...")
    model = pickle.load(open('fdemand.pkl', 'rb'))
    input_features = [float(x) for x in request.form.values()]
    features_value = [np.array(input_features)]
    print(features_value)

    features_name = ['homepage_featured', 'emailer_for_promotion', 'op_area', 'cuisine',
       'city_code', 'region_code', 'category']
    prediction = model.predict(features_value)
    output=prediction[0]
    print(output)
    return render_template('upload.html', prediction_text=output)


if__name__== '_main_':
     app.run(debug=False)
```

ibm.py

```
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY ="GC1ixTEle9bArXBEL10QAHEUIgEwD8Ni7ieLUfoP9oDc"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

# NOTE: manually define and pass the array(s) of values to be scored in the next line
payload_scoring = {"input_data": [{"fields": [array_of_input_fields], "values":
[array_of_values_to_be_scored, another_array_of_values_to_be_scored]}]}

response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/23fea940-01d1-4371-ab
d4-125cf59f5023/predictions?version=2022-11-18', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
print("Scoring response")
print(response_scoring.json())
predictions =response_scoring.json()
print(predictions)
print('Final Prediction Result',predictions['predictions'][0]['values'][0][0])
```

ibmapp.py

```
# import the necessary packages
import pandas as pd
import numpy as np
import pickle
import os
import requests

# NOTE: you must manually set API_KEY below using information retrieved from your IBM Cloud account.
API_KEY = "GC1ixTEle9bArXBEL10QAHEUIgEwD8Ni7ieLUfoP9oDc"
token_response = requests.post('https://iam.cloud.ibm.com/identity/token', data={"apikey":
 API_KEY, "grant_type": 'urn:ibm:params:oauth:grant-type:apikey'})
mltoken = token_response.json()["access_token"]
```

```
header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + mltoken}

from flask import Flask,request, render_template
app=Flask(_name_,template_folder="templates")
@app.route('/', methods=['GET'])
def index():
    return  render_template('home.html')
@app.route('/home', methods=['GET'])
def about():
    return render_template('home.html')
@app.route('/pred',methods=['GET'])
def page():
    return render_template('upload.html')
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    print("[INFO] loading model...")
    #model = pickle.load(open('fdemand.pkl', 'rb'))
    input_features = [int(x) for x in request.form.values()]
    print(input_features)
    features_value = [[np.array(input_features)]]
    print(features_value)

    payload_scoring = {"input_data":[{"field": [['homepage_featured', 'emailer_for_promotion',
'op_area', 'cuisine',
       'city_code', 'region_code', 'category']],"values": [input_features]}]}
    response_scoring =
requests.post('https://us-south.ml.cloud.ibm.com/ml/v4/deployments/23fea940-01d1-4371-ab
d4-125cf59f5023/predictions?version=2022-11-18', json=payload_scoring,
headers={'Authorization': 'Bearer ' + mltoken})
    print("Scoring response")
    print(response_scoring.json())
    predictions =response_scoring.json()
    print(predictions)
    print('Final Prediction Result',predictions['predictions'][0]['values'][0][0])
    pred = predictions['predictions'][0]['values'][0][0]

    #prediction = model.predict(features_value)
    #output=prediction[0]
    #print(output)
    print(pred)
    return render_template('upload.html', prediction_text=pred)


if__name__== '_main_':
     app.run(debug=False)
```