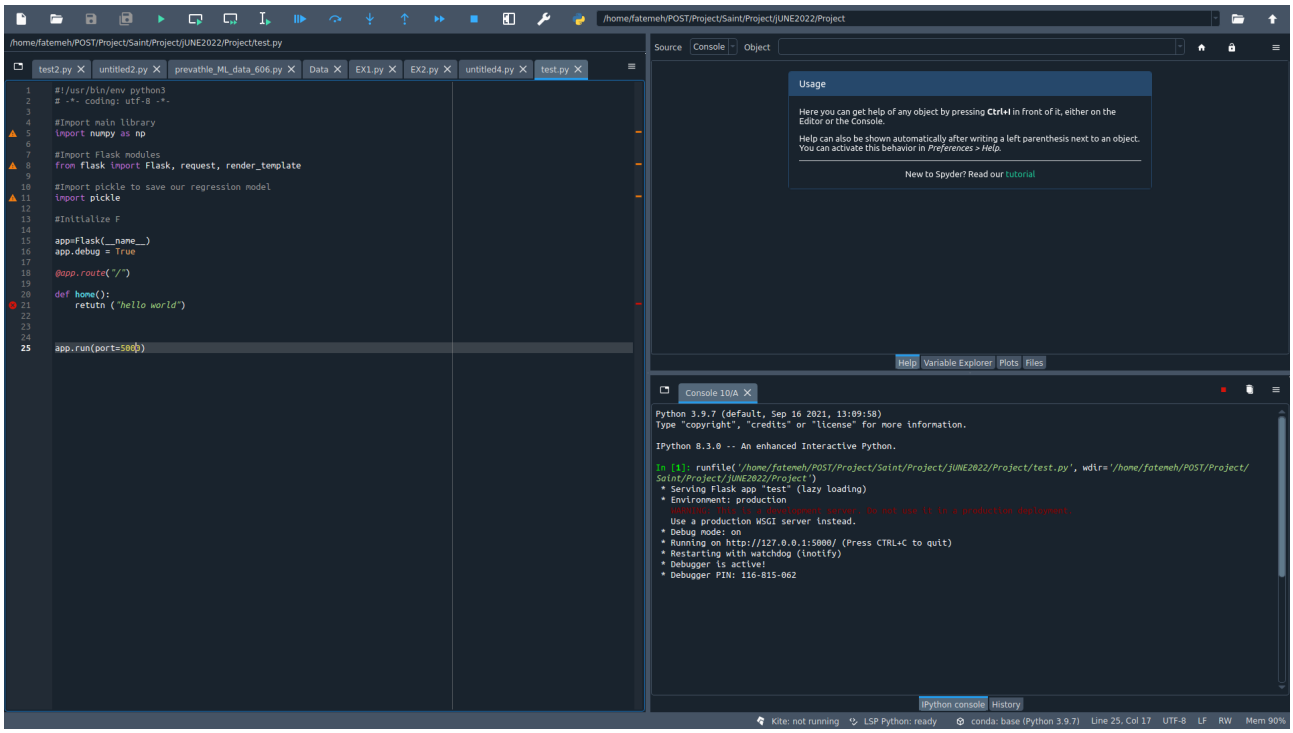
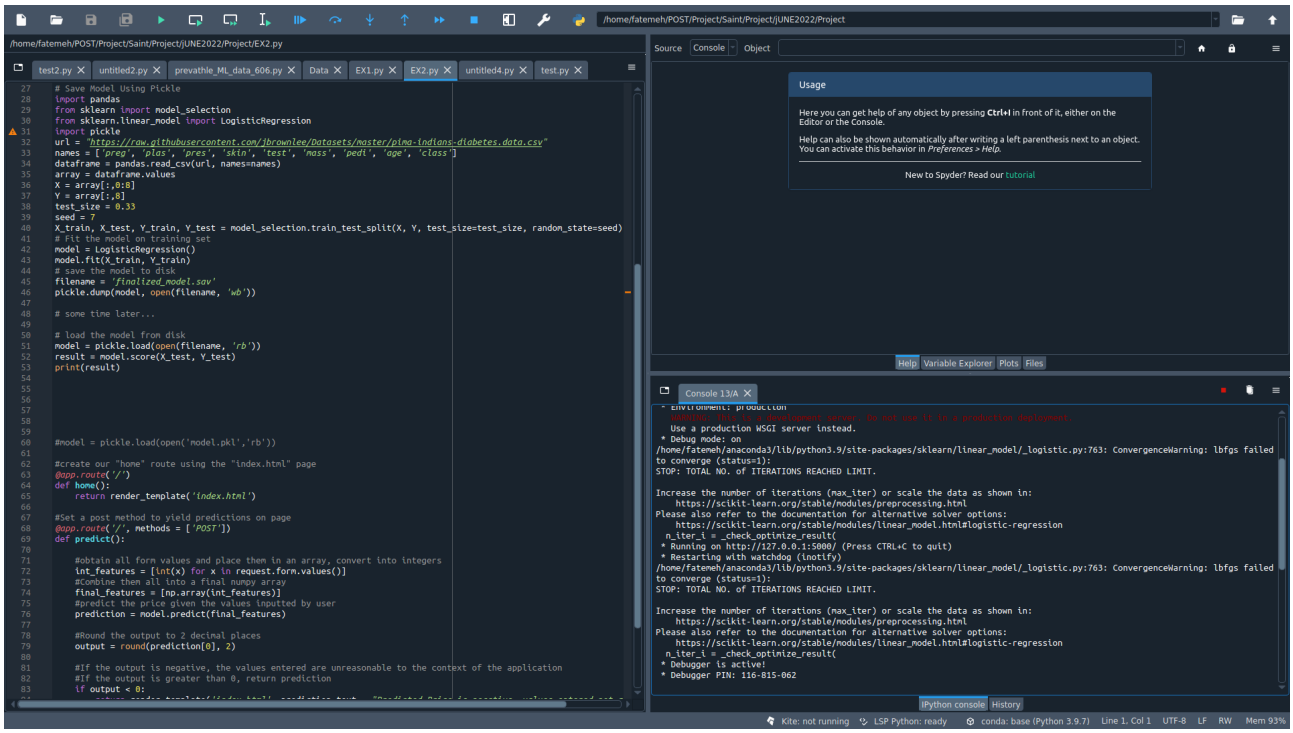
LISUM10: 30

Week4: Deployment on Flask

Fatemeh Bagheri

June2022




/home/fatemeh/POST/Project/Saint/Project/jUNE2022/Project
/home/fatemeh/POST/Project/Saint/Project/jUNE2022/Project/EX2.py
test2.py
untitled2.py
prevathle_ML_data_606.py
Data
EX1.py
EX2.py
untitled4.py
test.py
# Save Model Using Pickle
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
import pickle
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size, random_state=seed)
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))
# some time later...
# load the model from disk
model = pickle.load(open(filename, 'rb'))
result = model.score(X_test, Y_test)
print(result)
#model = pickle.load(open('model.pkl','rb'))
#create our "home" route using the "index.html" page
@app.route('/')
def home():
return render_template('index.html')
#Set a post method to yield predictions on page
@app.route('/', methods = ['POST'])
def predict():
#obtain all form values and place them in an array, convert into integers
int_features = [int(x) for x in request.form.values()]
#Combine them all into a final numpy array
final_features = [np.array(int_features)]
#predict the price given the values inputted by user
prediction = model.predict(final_features)
#Round the output to 2 decimal places
output = round(prediction[0], 2)
#If the output is negative, the values entered are unreasonable to the context of the application
#If the output is greater than 0, return prediction
if output < 0:
Source
Console
Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in Preferences > Help.
New to Spyder? Read our tutorial
Help
Variable Explorer
Plots
Files
Console 13/A
* 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
/home/fatemeh/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with watchdog (inotify)
/home/fatemeh/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
Increase the number of iterations (max_iter) or scale the data as shown in:
https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
n_iter_i = _check_optimize_result(
* Debugger is active!
* Debugger PIN: 116-815-062
IPython console
History
Kite: not running
LSP Python: ready
conda: base (Python 3.9.7)
Line 1, Col 1
UTF-8
LF
RW
Mem 93%

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Jun 29 17:02:38 2022

@author: fatemeh
"""

# -*- coding: utf-8 -*-



#Import main library
import numpy as np

#Import Flask modules
from flask import Flask, request, render_template

#Import pickle to save our regression model
import pickle

#Initialize Flask and set the template folder to "template"
app = Flask(__name__, template_folder = 'template')

#Open our model



# Save Model Using Pickle
import pandas
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
import pickle
```

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:,0:8]
Y = array[:,8]
test_size = 0.33
seed = 7
X_train, X_test, Y_train, Y_test = model_selection.train_test_split(X, Y, test_size=test_size, random_state=seed)
# Fit the model on training set
model = LogisticRegression()
model.fit(X_train, Y_train)
# save the model to disk
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))

# some time later...

# load the model from disk
model = pickle.load(open(filename, 'rb'))
result = model.score(X_test, Y_test)
print(result)
```

```
#model = pickle.load(open('model.pkl','rb'))
```

```
#create our "home" route using the "index.html" page
@app.route('/')
def home():
    return render_template('index.html')

#Set a post method to yield predictions on page
@app.route('/', methods = ['POST'])
def predict():

    #obtain all form values and place them in an array, convert into integers
    int_features = [int(x) for x in request.form.values()]
    #Combine them all into a final numpy array
    final_features = [np.array(int_features)]
    #predict the price given the values inputted by user
    prediction = model.predict(final_features)

    #Round the output to 2 decimal places
    output = round(prediction[0], 2)

    #If the output is negative, the values entered are unreasonable to the context of the application
    #If the output is greater than 0, return prediction
    if output < 0:
        return render_template('index.html', prediction_text = "Predicted Price is negative, values entered not reasonable")
    elif output >= 0:
        return render_template('index.html', prediction_text = 'Predicted Price of the house is: $ {}'.format(output))

#Run app
if __name__ == "__main__":
    app.run(debug=True)
```