

Name: Nathan Adam

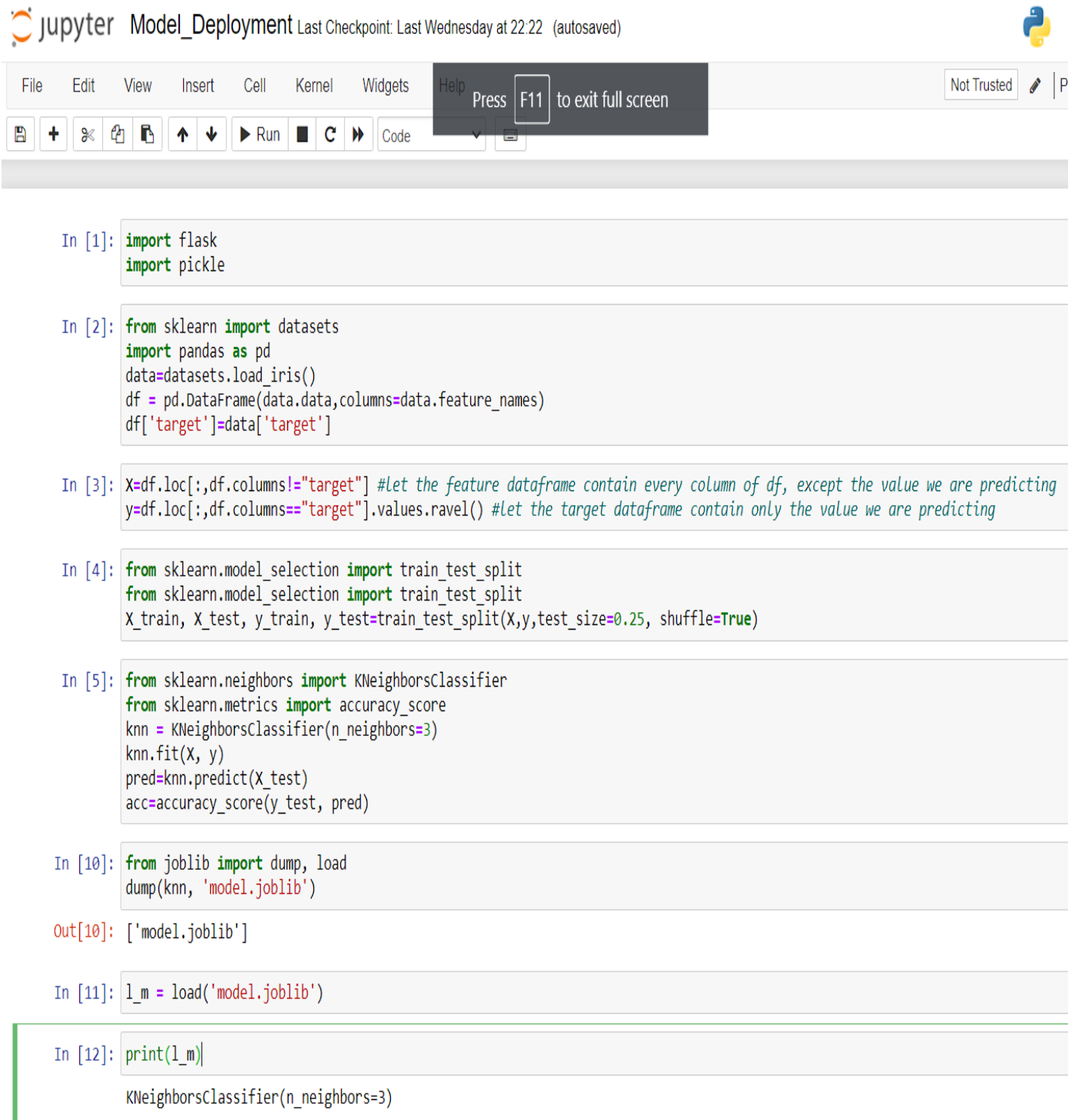
Batch code: LISUM01

Submission date: 3<sup>rd</sup> July 2021

Submitted to: Week 4 Deployment on Flask (on canvas)

[https://github.com/N-A-ML/Data\\_Glacier\\_Deployment\\_on\\_Flask\\_Week\\_4](https://github.com/N-A-ML/Data_Glacier_Deployment_on_Flask_Week_4) (on GitHub)

Select data (iris dataset), create and save a simple model (knn classifier):



The image shows a Jupyter Notebook interface with the title 'Model\_Deployment'. The top bar includes the Jupyter logo, the title, and a status message 'Last Checkpoint: Last Wednesday at 22:22 (autosaved)'. On the right, there is a Python logo and a 'Not Trusted' warning. Below the title bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Widgets'. A tooltip for the 'Help' menu is visible, showing 'Press F11 to exit full screen'. Below the menu bar is a toolbar with icons for saving, adding cells, running, and other standard Jupyter actions. The main area of the notebook contains 12 code cells. The first cell imports 'flask' and 'pickle'. The second cell imports 'sklearn.datasets', 'pandas', and loads the 'iris' dataset into a DataFrame. The third cell splits the data into features (X) and targets (y). The fourth cell uses 'train\_test\_split' to create training and testing sets. The fifth cell trains a 'KNeighborsClassifier' and evaluates its accuracy. The sixth cell saves the trained model using 'joblib.dump'. The seventh cell shows the output of the save operation. The eighth cell loads the saved model. The ninth cell prints the loaded model object.

```
In [1]: import flask
import pickle

In [2]: from sklearn import datasets
import pandas as pd
data=datasets.load_iris()
df = pd.DataFrame(data.data,columns=data.feature_names)
df['target']=data['target']

In [3]: X=df.loc[:,df.columns!="target"] #let the feature dataframe contain every column of df, except the value we are predicting
y=df.loc[:,df.columns=="target"].values.ravel() #let the target dataframe contain only the value we are predicting

In [4]: from sklearn.model_selection import train_test_split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.25, shuffle=True)

In [5]: from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X, y)
pred=knn.predict(X_test)
acc=accuracy_score(y_test, pred)

In [10]: from joblib import dump, load
dump(knn, 'model.joblib')

Out[10]: ['model.joblib']

In [11]: l_m = load('model.joblib')

In [12]: print(l_m)

KNeighborsClassifier(n_neighbors=3)
```

## Create html and css files:

index.html - Notepad

File Edit Format View Help

```
<html>
<head>
<link rel= "stylesheet" type= "text/css" href= "{{ url_for('static',filename='styles/styles.css') }}">
<link rel="stylesheet" type="text/css" href="//fonts.googleapis.com/css?family=Playfair+Display" />
<title> Predict the type of iris flower </title>
</head>

<body>
<h1> Predict the type of iris flower (Setosa, Versicolor, or Virginica) using a K nearest neighbors classifier (k=3)</h1>
<div class="wrapper">
<div class="form">

<form action = "{{ url_for('predict')}}" method="post">
    <input type="text" name="sepal_length" placeholder= "Sepal Length(cm)" required="required" /> <br> <br>
    <input type="text" name="sepal_width" placeholder= "Sepal Width(cm)" required="required" /> <br> <br>
    <input type="text" name="petal_length" placeholder= "Petal Length(cm)" required="required" /> <br> <br>
    <input type="text" name="petal_width" placeholder= "Petal Width(cm)" required="required" /> <br> <br>
    <button type="submit"> Predict </button>

|
<br>
<br>
{{ prediction_text }}
</form>

</div>
    <div class="image">
    
    </div>

</div>
</body>
</html>
```

\*styles.css - Notepad

File Edit Format View Help

```
* {
    font-family:"Playfair Display";
}
body {
    background-color: lightblue;
}


h1 {
font-size:3.5em;
margin-left:5%;
margin-right:5%;
}

form input, button {
font-size:1.5em;
}
form {
font-size:1.5em;
}

.wrapper {
display:flex;
justify-content:space-evenly;
margin-top:5%;
}

.form {
}
|
.image img{
    max-height:50vh;
    height:auto;
    width:auto;
}
}
```

Use flask so the web app can be deployed locally. Images are included in the app:












 jupyter Deployment\_flask.py a few seconds ago

File Edit View Language

Press **F11** to exit full screen

```
1  #!/usr/bin/env python
2  # coding: utf-8
3
4  # In[ ]:
5  import numpy as np
6  from flask import Flask, request, render_template
7  import joblib
8  from joblib import load
9  from sklearn.neighbors import KNeighborsClassifier
10 import os
11 images_folder=os.path.join('static', 'images')
12 app=Flask(__name__)
13 app.config['UPLOAD_FOLDER'] = images_folder
14 model=load('model.joblib')
15
16 @app.route('/')
17 def home():
18     return render_template('index.html')
19 @app.route('/predict', methods=['POST'])
20 def predict():
21     features=[float(x) for x in request.form.values()]
22     final_features=[np.array(features)]
23     prediction=model.predict(final_features)
24     pred_round=round(prediction[0])
25     output=""
26     if pred_round==0:
27         output+="Setosa"
28         file = os.path.join(app.config['UPLOAD_FOLDER'], 'setosa.jpg')
29     elif pred_round==1:
30         output+="Versicolor"
31         file = os.path.join(app.config['UPLOAD_FOLDER'], 'versicolor.jpg')
32     else:
33         output+="Virginica"
34         file = os.path.join(app.config['UPLOAD_FOLDER'], 'virginica.jpg')
35
36     return render_template('index.html', prediction_text='This iris flower is {}'.format(output),
37                           iris=file
38                           )
39 if __name__ == "__main__":
40     app.run(port=5000, debug=True, use_reloader=False)
41 |
42 # In[16]:
43
```

Generate Procfile (and enter the name of the app), requirements.txt and runtime.txt, and structure the files and folders correctly:

Add folder in library ▾		Give access to ▾	New folder	
Name		Date modified	Type	Size
 .git		02/07/2021 19:14	File folder	
 .ipynb_checkpoints		30/06/2021 20:30	File folder	
 static		02/07/2021 17:04	File folder	
 templates		30/06/2021 21:58	File folder	
 Deployment_flask.ipynb		02/07/2021 19:30	IPYNB File	5 KB
 Deployment_flask.py		02/07/2021 19:30	Python File	2 KB
 model.joblib		30/06/2021 22:08	JOBLIB File	14 KB
 Model_Deployment.ipynb		30/06/2021 22:22	IPYNB File	7 KB
 Procfile		01/07/2021 17:28	File	1 KB
 requirements.txt		01/07/2021 18:00	Text Document	1 KB
 runtime.txt		01/07/2021 17:29	Text Document	1 KB

Ensure that relevant packages are installed in the working directory (e.g., gunicorn) and upload the files and folders to GitHub. Link the GitHub repository to Heroku and troubleshoot any problems by checking the logs.

Salesforce Platform

HEROKU

Jump to Favorites, Apps, Pipelines, Spaces...

Personal



>

predict3iris

☆

Open app

More


GitHub  N-A-ML/Data\_Glacier\_Deployment\_on\_Flask\_Week\_4  main

Overview Resources **Deploy** Metrics Activity Access Settings

Add this app to a pipeline


Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features



Pipelines let you connect multiple apps together and **promote code** between them.

[Learn more.](#)





Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests.


[Learn more.](#)

Choose a pipeline

Deployment method



 Heroku Git  
Use Heroku CLI

 GitHub  
Connected


 Container Registry  
Use Heroku CLI



App connected to GitHub

Code diffs, manual and auto deploys are available for this app.


Connected to  N-A-ML/Data\_Glacier\_Deployment\_on\_Flask\_Week\_4 by  N-A-ML 

Disconnect...

 Releases in the [activity feed](#) link to GitHub to view commit diffs

 Automatically deploys from  main

Automatic deploys

 You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please

Finally, launch the app and test it:

**Predict the type of iris flower (Setosa, Versicolor, or Virginica) using a K nearest neighbors classifier (k=3)**

Sepal Length(cm)


Sepal Width(cm)

Petal Length(cm)

Petal Width(cm)

Predict

This iris flower is Setosa



The app is working as intended.