Name: Nathan Adam

Batch code: LISUM01
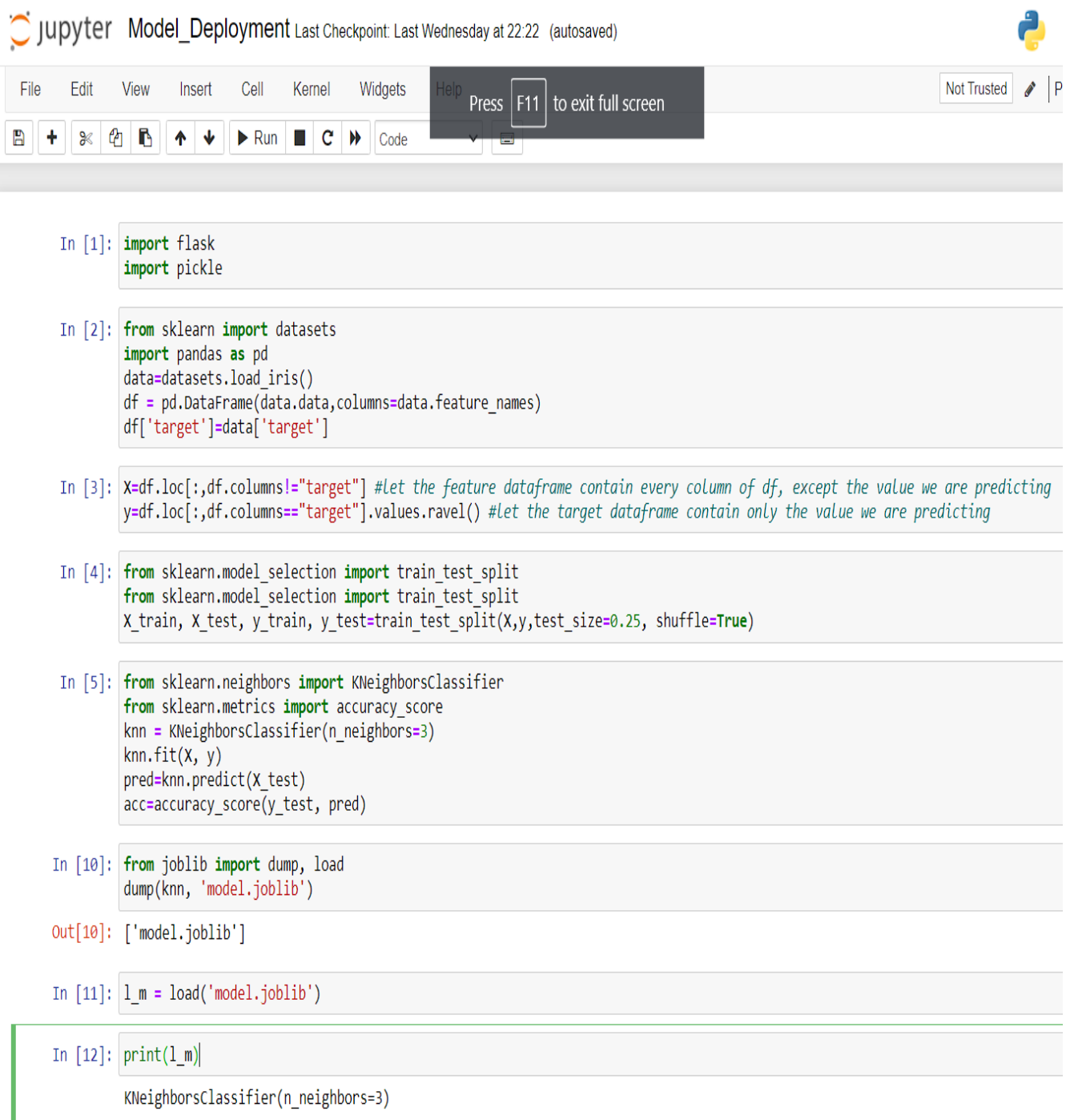
Submission date:7th July 2021

Submitted to: Week 5: Cloud and API deployment

https://github.com/N-A-ML/Data_Glacier_Cloud_and_API_Deployment_Week_5 (on GitHub)

Note: the app was deployed on the cloud with Heroku in week 4 but is documented here also.

App URL: https://predict3iris.herokuapp.com/

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


```python
import flask
import pickle
```

```python
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']
```

```python
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
```

```python
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)
```

```python
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)
```

```python
from joblib import dump, load
dump(knn, 'model.joblib')
```

```
['model.joblib']
```

```python
l_m = load('model.joblib')
```

```python
print(l_m)
```

```
KNeighborsClassifier(n_neighbors=3)
```

Create html and css files:

```html
<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">
 <img src="{{ iris }}" alt="">
        </div>

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

```css
* {
 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:

File   Edit   View   Language

Press F11 to exit full screen

```python
#!/usr/bin/env python
# coding: utf-8

# In[ ]:
import numpy as np
from flask import Flask, request, render_template
import joblib
from joblib import load
from sklearn.neighbors import KNeighborsClassifier
import os
images_folder=os.path.join('static', 'images')
app=Flask(__name__)
app.config['UPLOAD_FOLDER'] = images_folder
model=load('model.joblib')

@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    features=[float(x) for x in request.form.values()]
    final_features=[np.array(features)]
    prediction=model.predict(final_features)
    pred_round=round(prediction[0])
    output=""
    if pred_round==0:
        output+="Setosa"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'setosa.jpg')
    elif pred_round==1:
        output+="Versicolor"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'versicolor.jpg')
    else:
        output+="Virginica"
        file = os.path.join(app.config['UPLOAD_FOLDER'], 'virginica.jpg')

    return render_template('index.html', prediction_text='This iris flower is {}'.format(output),
                        iris=file
                        )
if __name__=="__main__":
    app.run(port=5000, debug=True, use_reloader=False)

# In[16]:
```

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

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.

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

The app is working as intended.

Each type of prediction was tested using Postman:

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 Versicolor

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 Virginica

The iris types were predicted correctly and everything is working as intended.