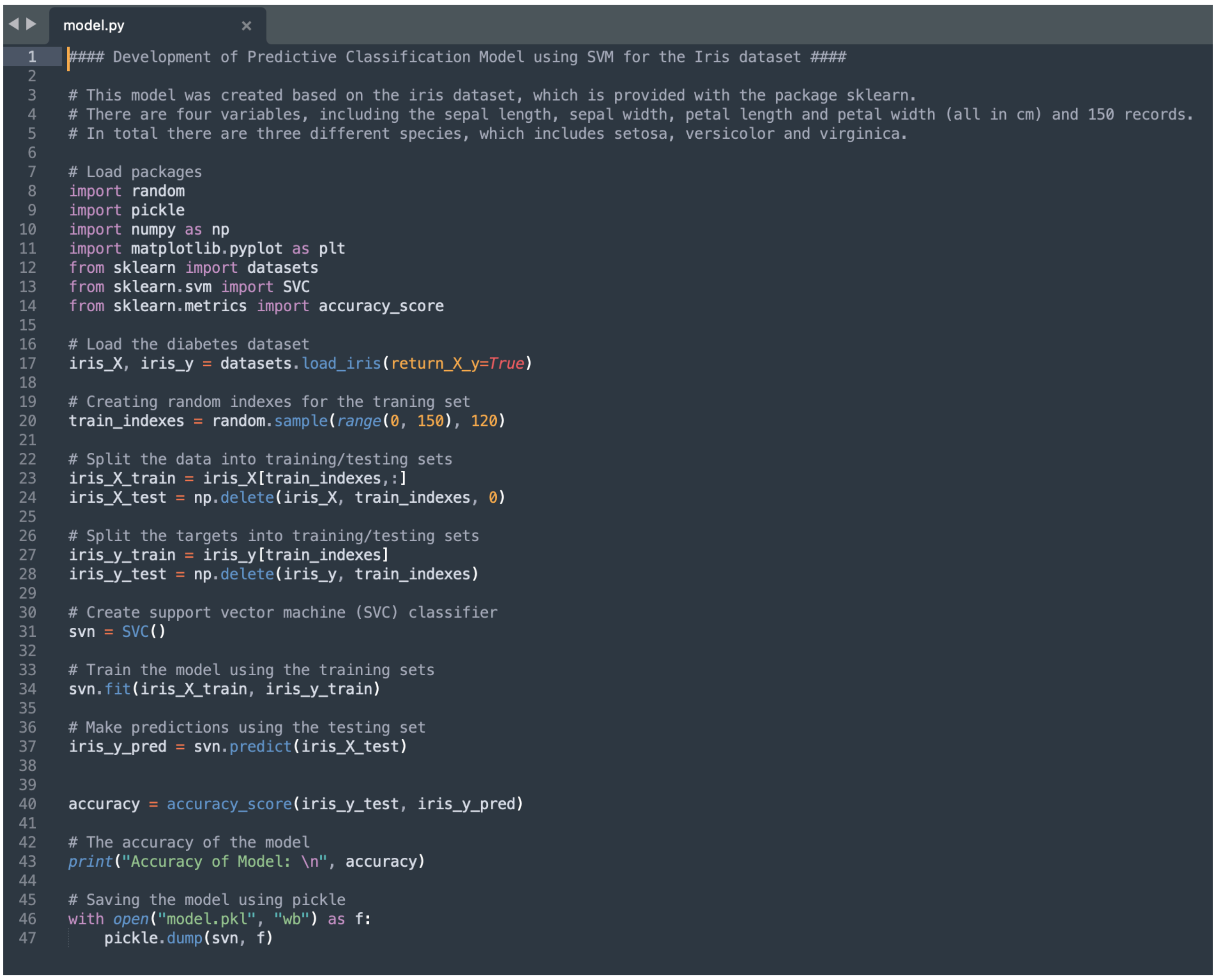
Name: **Mario Rodrigues Peres**
Batch code: **LISUM19**
Submission date: **March 22nd, 2023**
Submitted to: **Glacier Data Scientist Internship**

STEPS FOR FLASK DEPLOYMENT OF ML MODEL

1. Creation of a simple linear regression model, which was saved as model.pkl:

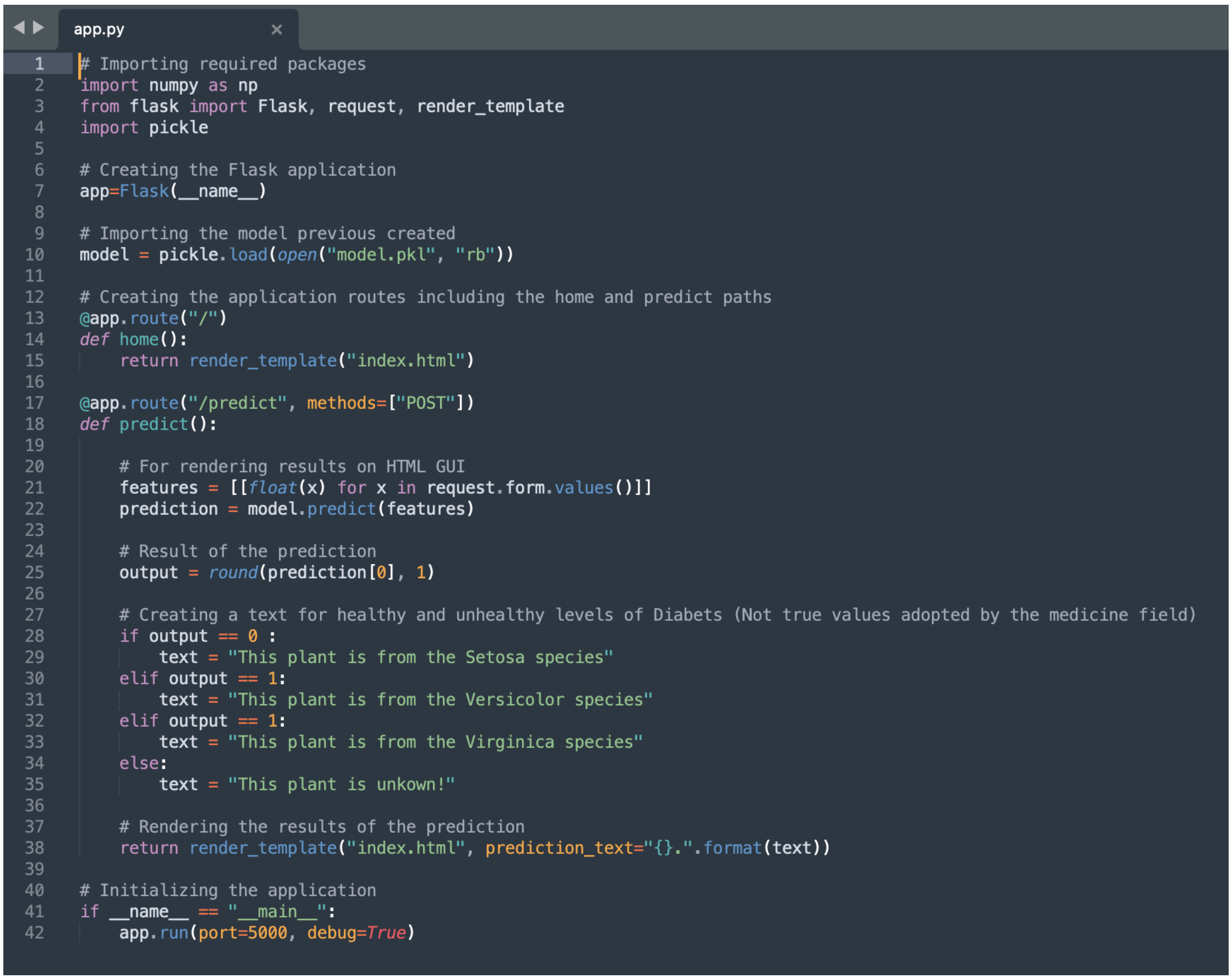


model.py

```python
#### Development of Predictive Classification Model using SVM for the Iris dataset ####

# This model was created based on the iris dataset, which is provided with the package sklearn.
# There are four variables, including the sepal length, sepal width, petal length and petal width (all in cm) and 150 records.
# In total there are three different species, which includes setosa, versicolor and virginica.

# Load packages
import random
import pickle
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

# Load the diabetes dataset
iris_X, iris_y = datasets.load_iris(return_X_y=True)

# Creating random indexes for the traning set
train_indexes = random.sample(range(0, 150), 120)

# Split the data into training/testing sets
iris_X_train = iris_X[train_indexes,:]
iris_X_test = np.delete(iris_X, train_indexes, 0)

# Split the targets into training/testing sets
iris_y_train = iris_y[train_indexes]
iris_y_test = np.delete(iris_y, train_indexes)

# Create support vector machine (SVC) classifier
svn = SVC()

# Train the model using the training sets
svn.fit(iris_X_train, iris_y_train)

# Make predictions using the testing set
iris_y_pred = svn.predict(iris_X_test)


accuracy = accuracy_score(iris_y_test, iris_y_pred)

# The accuracy of the model
print("Accuracy of Model: \n", accuracy)

# Saving the model using pickle
with open("model.pkl", "wb") as f:
    pickle.dump(svn, f)
```

In this first step, the dataset Iris from the Sklearn was used as the toy data to create a ML model using the support vector machine applied to classification. First the dataset is loaded and randomly separated into train and test datasets. Following that an instance of the SVC function was used to create an instance of the model, which was fitted with the train data. After the training, the model was tested with the test data and the accuracy calculated. Finally, the model was serialized and saved to be used in the web app.

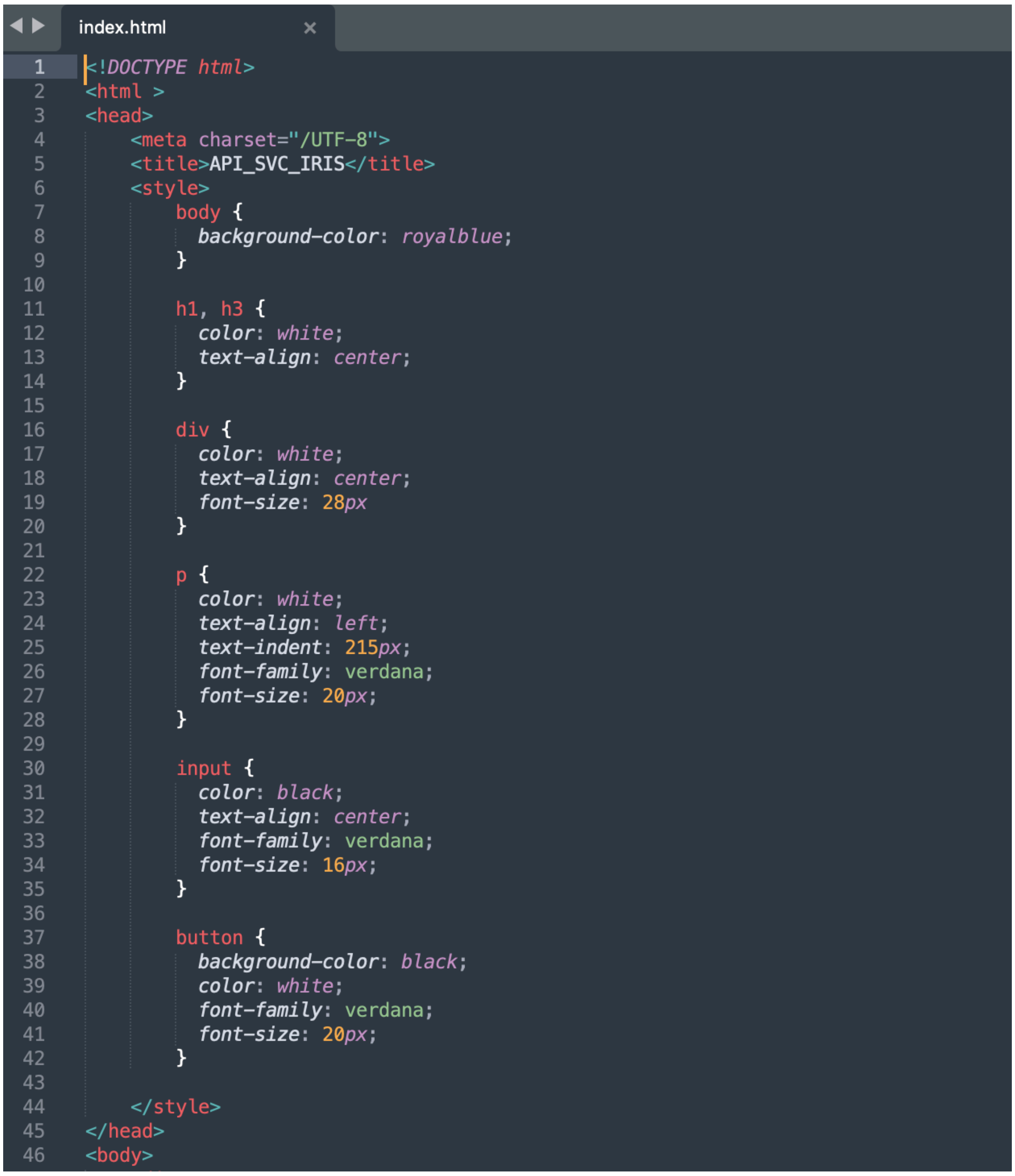
2. Creation of the “app.py” script:



```python
# Importing required packages
import numpy as np
from flask import Flask, request, render_template
import pickle

# Creating the Flask application
app=Flask(__name__)

# Importing the model previous created
model = pickle.load(open("model.pkl", "rb"))

# Creating the application routes including the home and predict paths
@app.route("/")
def home():
    return render_template("index.html")

@app.route("/predict", methods=["POST"])
def predict():

    # For rendering results on HTML GUI
    features = [[float(x) for x in request.form.values()]]
    prediction = model.predict(features)

    # Result of the prediction
    output = round(prediction[0], 1)

    # Creating a text for healthy and unhealthy levels of Diabets (Not true values adopted by the medicine field)
    if output == 0 :
        text = "This plant is from the Setosa species"
    elif output == 1:
        text = "This plant is from the Versicolor species"
    elif output == 1:
        text = "This plant is from the Virginica species"
    else:
        text = "This plant is unkown!"

    # Rendering the results of the prediction
    return render_template("index.html", prediction_text="{}.".format(text))

# Initializing the application
if __name__ == "__main__":
    app.run(port=5000, debug=True)
```

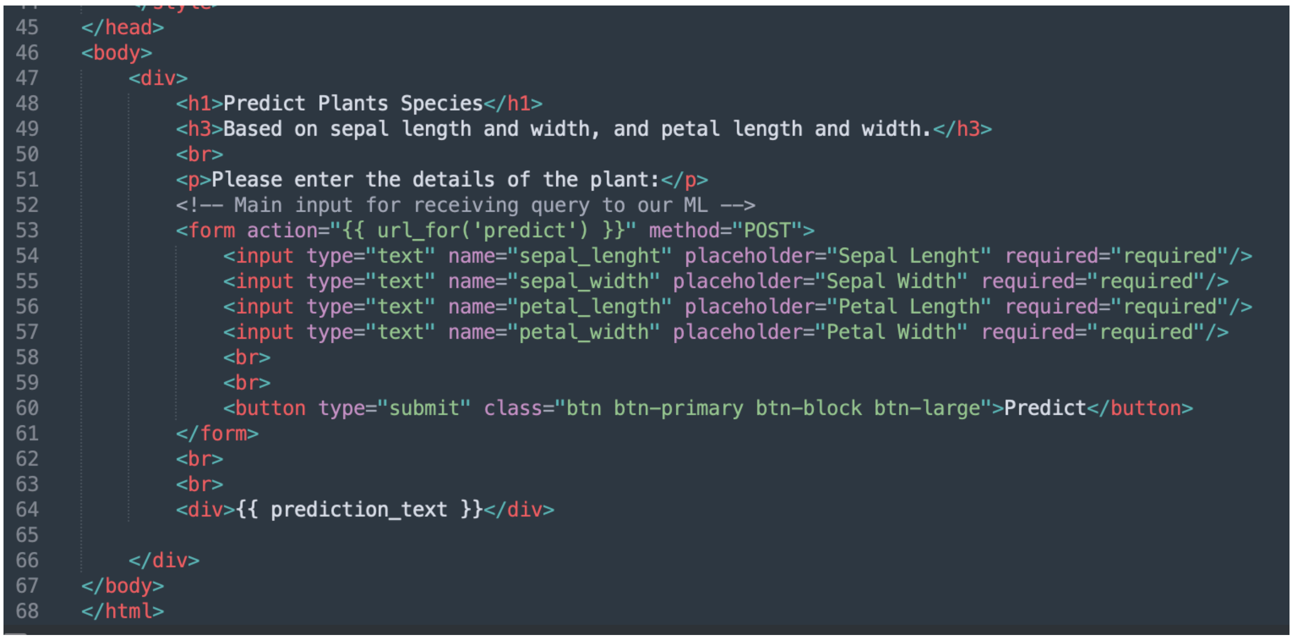
The app.py file is the application. First it was created an instance of the Flask and the model was loaded. After that, the routes for the home and predict pages were created. There was also created a text that shows the specie of plant that the model predicted.

3. Creation of the “index.html” script:



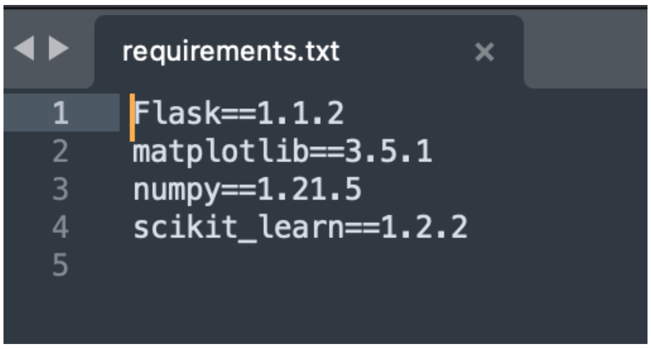
```
index.html
<!DOCTYPE html>
<html >
<head>
    <meta charset="/UTF-8">
    <title>API_SVC_IRIS</title>
    <style>
        body {
          background-color: royalblue;
        }

        h1, h3 {
          color: white;
          text-align: center;
        }

        div {
          color: white;
          text-align: center;
          font-size: 28px
        }

        p {
          color: white;
          text-align: left;
          text-indent: 215px;
          font-family: verdana;
          font-size: 20px;
        }

        input {
          color: black;
          text-align: center;
          font-family: verdana;
          font-size: 16px;
        }

        button {
          background-color: black;
          color: white;
          font-family: verdana;
          font-size: 20px;
        }

    </style>
</head>
<body>
```

The index.html file is very simple. First, it was created a title for the web page and the style for the different sections of the page (figure above). Following that, there were created a few elements with text, such as the headings 1 and 3, paragraph and a form where the user can input values for the sepal length and width as well as petal length and width. Moreover, it was created a submit button to allow the user to make the prediction, which is displayed in the /predict page (next figure).



```html
</head>
<body>
    <div>
        <h1>Predict Plants Species</h1>
        <h3>Based on sepal length and width, and petal length and width.</h3>
        <br>
        <p>Please enter the details of the plant:</p>
        <!-- Main input for receiving query to our ML -->
        <form action="{{ url_for('predict') }}" method="POST">
            <input type="text" name="sepal_lenght" placeholder="Sepal Lenght" required="required"/>
            <input type="text" name="sepal_width" placeholder="Sepal Width" required="required"/>
            <input type="text" name="petal_length" placeholder="Petal Length" required="required"/>
            <input type="text" name="petal_width" placeholder="Petal Width" required="required"/>
            <br>
            <br>
            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
        </form>
        <br>
        <br>
        <div>{{ prediction_text }}</div>

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

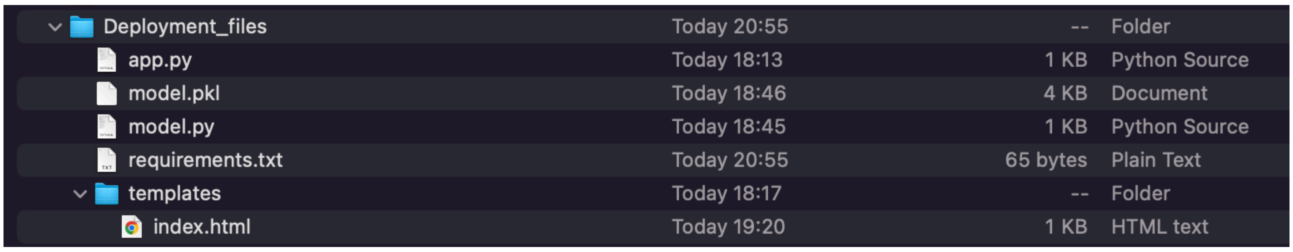
4. Creation of the requirements.txt:



```
Flask==1.1.2
matplotlib==3.5.1
numpy==1.21.5
scikit_learn==1.2.2
```

The requirement.txt was create using the package pipreqs, which automatically generates the requirement.txt with the packages used in the project.

5. Files and folders organization:



Deployment_files	Today 20:55	--	Folder
app.py	Today 18:13	1 KB	Python Source
model.pkl	Today 18:46	4 KB	Document
model.py	Today 18:45	1 KB	Python Source
requirements.txt	Today 20:55	65 bytes	Plain Text
templates	Today 18:17	--	Folder
index.html	Today 19:20	1 KB	HTML text

This last picture represents the directory holding all the files and folders used for the deployment of the web application. The directory templates holds the html files, whereas the main directory holds all other files, including the app.py, model.pkl, model.py and requirements.txt.