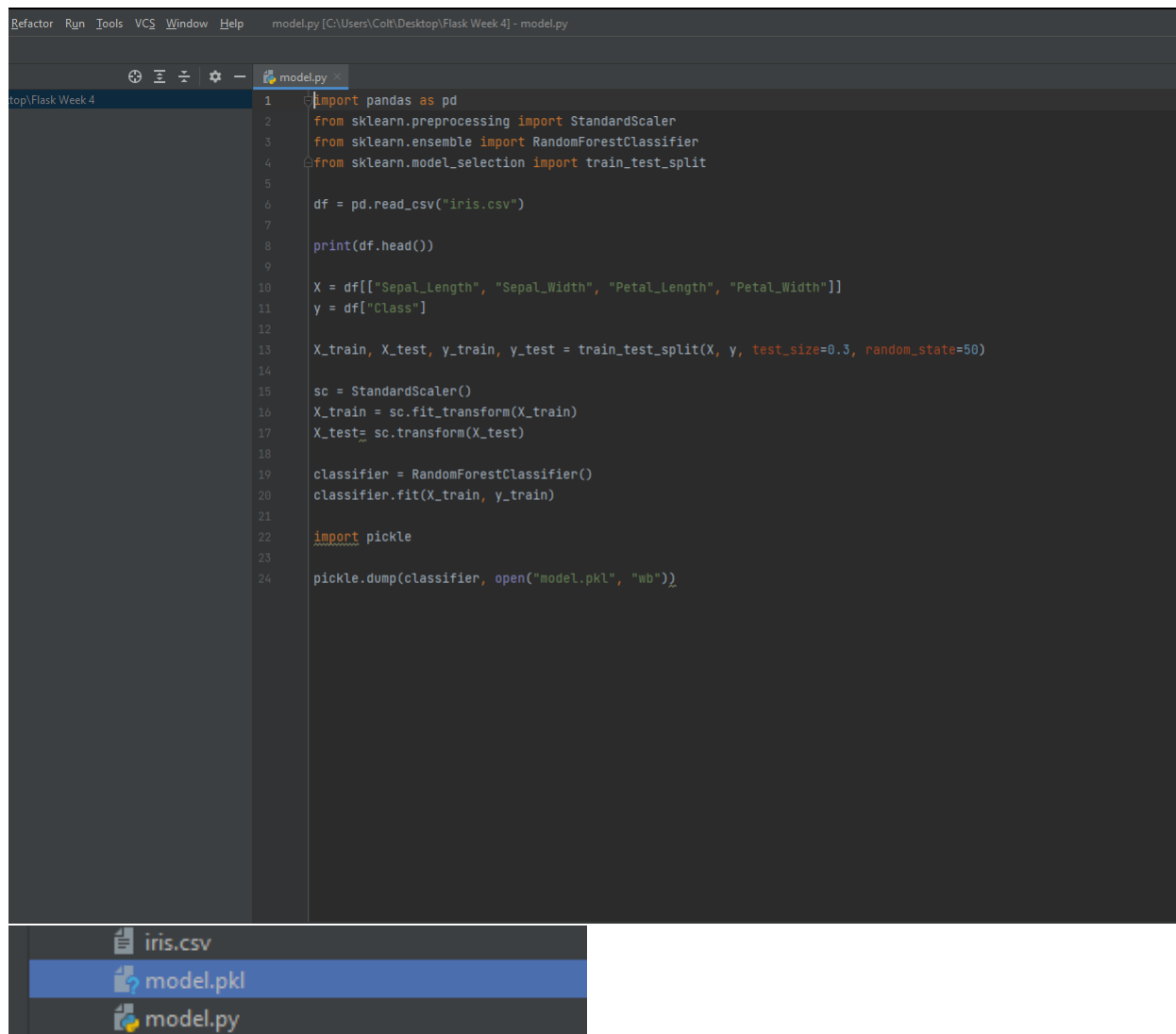


Name: Richard Coltenback

Batch Code: LISUM12

Report date: August 28th 2022

Data storage location: <https://github.com/RColtenback/Flask-Deployment-Week-4>



```
Refactor Run Tools VCS Window Help model.py [C:\Users\Colt\Desktop\Flask Week 4] - model.py

top\Flask Week 4
1 import pandas as pd
2 from sklearn.preprocessing import StandardScaler
3 from sklearn.ensemble import RandomForestClassifier
4 from sklearn.model_selection import train_test_split
5
6 df = pd.read_csv("iris.csv")
7
8 print(df.head())
9
10 X = df[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
11 y = df["Class"]
12
13 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)
14
15 sc = StandardScaler()
16 X_train = sc.fit_transform(X_train)
17 X_test = sc.transform(X_test)
18
19 classifier = RandomForestClassifier()
20 classifier.fit(X_train, y_train)
21
22 import pickle
23
24 pickle.dump(classifier, open("model.pkl", "wb"))
```

The screenshot shows a Python IDE with a file explorer on the left. The file explorer contains three files: 'iris.csv', 'model.pkl', and 'model.py'. The 'model.py' file is selected and highlighted in blue. The main editor area displays the Python code for training a RandomForestClassifier and saving it as a pickle file. The code includes imports for pandas, sklearn preprocessing, sklearn ensemble, and sklearn model selection. It reads the 'iris.csv' file, prints the first five rows, and splits the data into training and testing sets. A StandardScaler is used to scale the features. A RandomForestClassifier is trained on the training data and saved as 'model.pkl' using the pickle module.

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4
5 flask_app = Flask(__name__)
6 model = pickle.load(open("model.pkl", "rb"))
7
8 @flask_app.route("/")
9 def Home():
10     return render_template("index.html")
11
12 @flask_app.route("/predict", methods = ["POST"])
13 def predict():
14     float_features = [float(x) for x in request.form.values()]
15     features = [np.array(float_features)]
16     prediction = model.predict(features)
17     return render_template("index.html", prediction_text = "The flower species is {}".format(prediction))
18
19 if __name__ == "__main__":
20     flask_app.run(debug=True)
```

```
Run: app ×
warnings.warn
* Serving Flask app 'app' (lazy loading)
* 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
* Running on http://127.0.0.1:5000 (Press CTRL+C to quit)
```

←

→

🔄

🔍 127.0.0.1:5000

🔗

☆

🔴

🔵

🟢

🟠

🟡

🟣

⋮

🔍 Google

📺 YouTube

📺 AOL

📺 Amazon

📺 Netflix

📺 Hulu

📺 Disney+

📺 HBO Max

📺 Peacock

📺 STARZ

📺 SHOWTIME

📺 Vudu

📺 NFL.com

📺 Sportsurge

📺 New York Giants ES...

📺 League Office - Wal...

📺 FantasyPros

📺 Can You RUN It

📺 25 Jokes That Are S...

»

Predicting Flower Species

Sepal Length

Sepal Width

Petal Length

Petal Width

Predict

←

→

🔄

🔍 127.0.0.1:5000/predict

🔗

☆

🔴

🔵

🟢

🟠

🟡

🟣

⋮

🔍 Google

📺 YouTube

📺 AOL

📺 Amazon

📺 Netflix

📺 Hulu

📺 Disney+

📺 HBO Max

📺 Peacock

📺 STARZ

📺 SHOWTIME

📺 Vudu

📺 NFL.com

📺 Sportsurge

📺 New York Giants ES...

📺 League Office - Wal...

📺 FantasyPros

📺 Can You RUN It

📺 25 Jokes That Are S...

»

Predicting Flower Species

1

85

2

88

Predict

The flower species is ["Versicolor"]