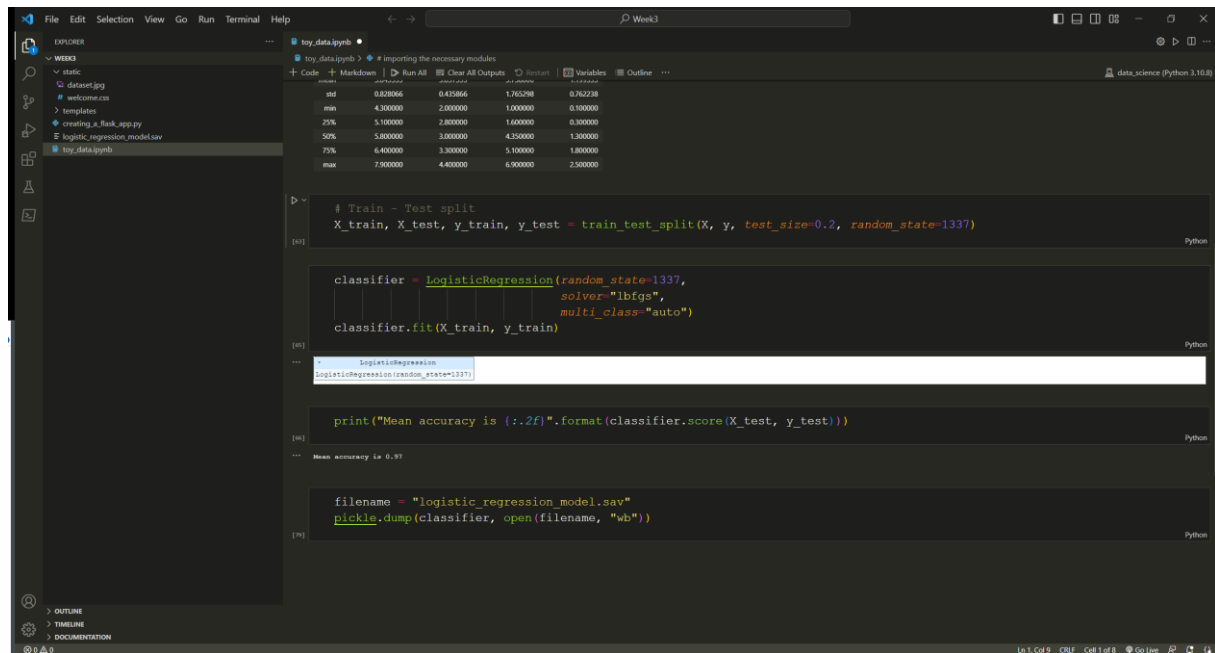


Halit Ayberk DEMIR

Batch code = LISUM19

h.ayberk.demir.34@gmail.com

1 – Creating a logistic regression model with iris dataset from scikit learn library



The screenshot shows a Jupyter Notebook with the following code and output:

```
# Importing the necessary modules
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Load the iris dataset
iris = load_iris()

# Train - Test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1337)

# Create a LogisticRegression object
classifier = LogisticRegression(random_state=1337, solver='lbfgs', multi_class='auto')

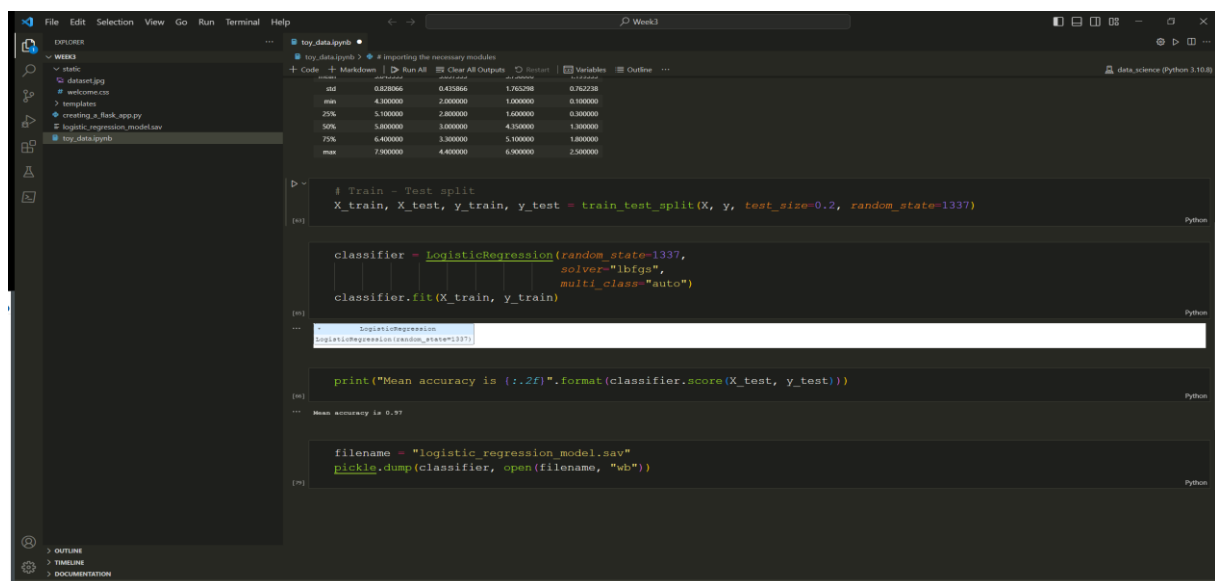
# Fit the model
classifier.fit(X_train, y_train)

# Predict on the test set
y_pred = classifier.predict(X_test)

# Calculate the mean accuracy
print("Mean accuracy is {:.2f}".format(classifier.score(X_test, y_test)))
```

The output shows the mean accuracy is 0.97.

2 – Fitting the model to dataset, testing the accuracy and saving the model with python's pickle library.



The screenshot shows a Jupyter Notebook with the following code and output:

```
# Importing the necessary modules
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pickle

# Load the iris dataset
iris = load_iris()

# Train - Test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1337)

# Create a LogisticRegression object
classifier = LogisticRegression(random_state=1337, solver='lbfgs', multi_class='auto')

# Fit the model
classifier.fit(X_train, y_train)

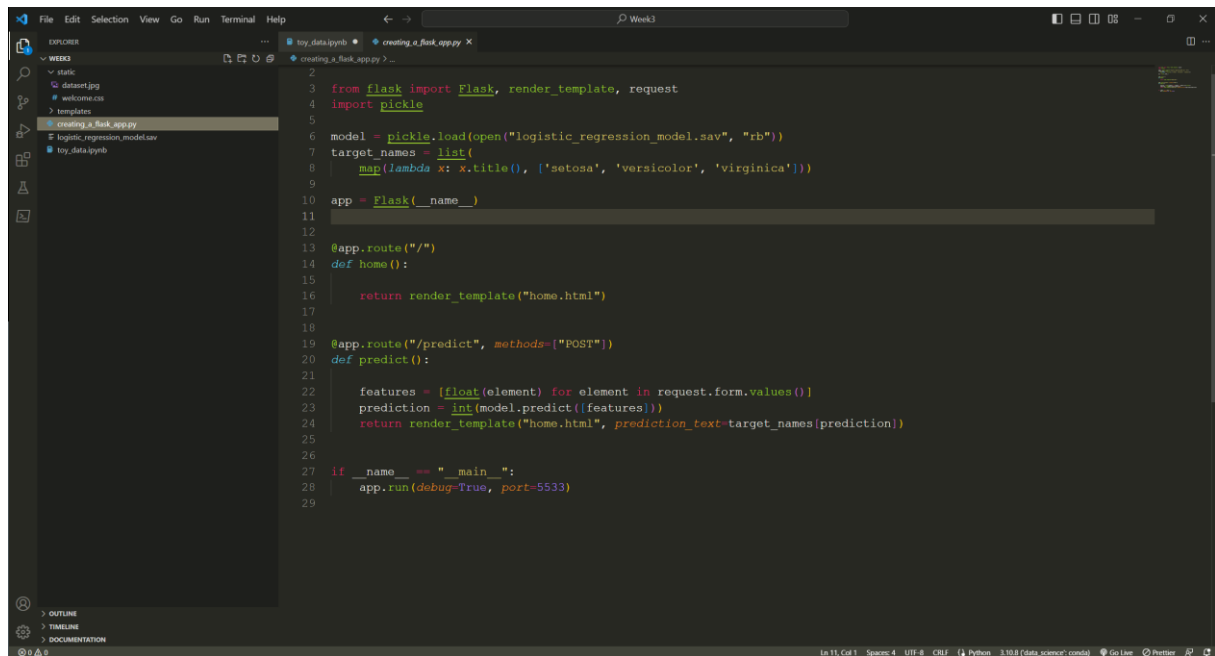
# Predict on the test set
y_pred = classifier.predict(X_test)

# Calculate the mean accuracy
print("Mean accuracy is {:.2f}".format(classifier.score(X_test, y_test)))

# Save the model
filename = "logistic_regression_model.sav"
pickle.dump(classifier, open(filename, "wb"))
```

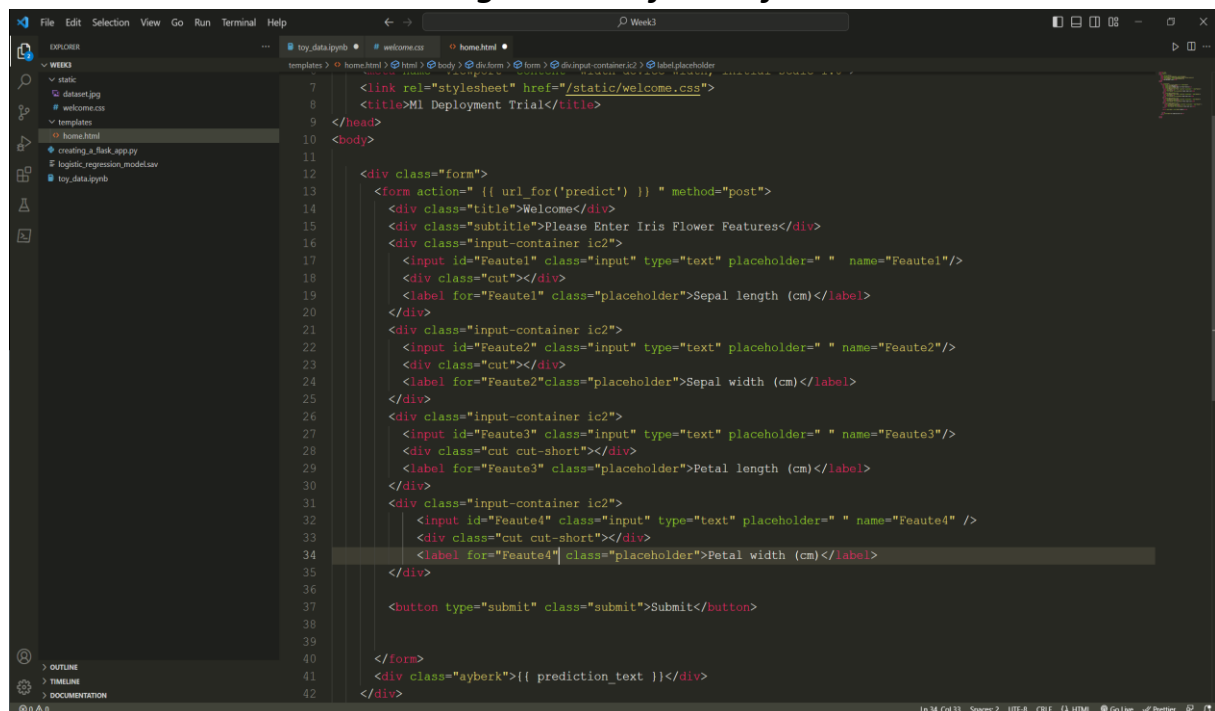
The output shows the mean accuracy is 0.97.

3 – Creating a Flask app, loading the pickle model and applying the model at website's backend



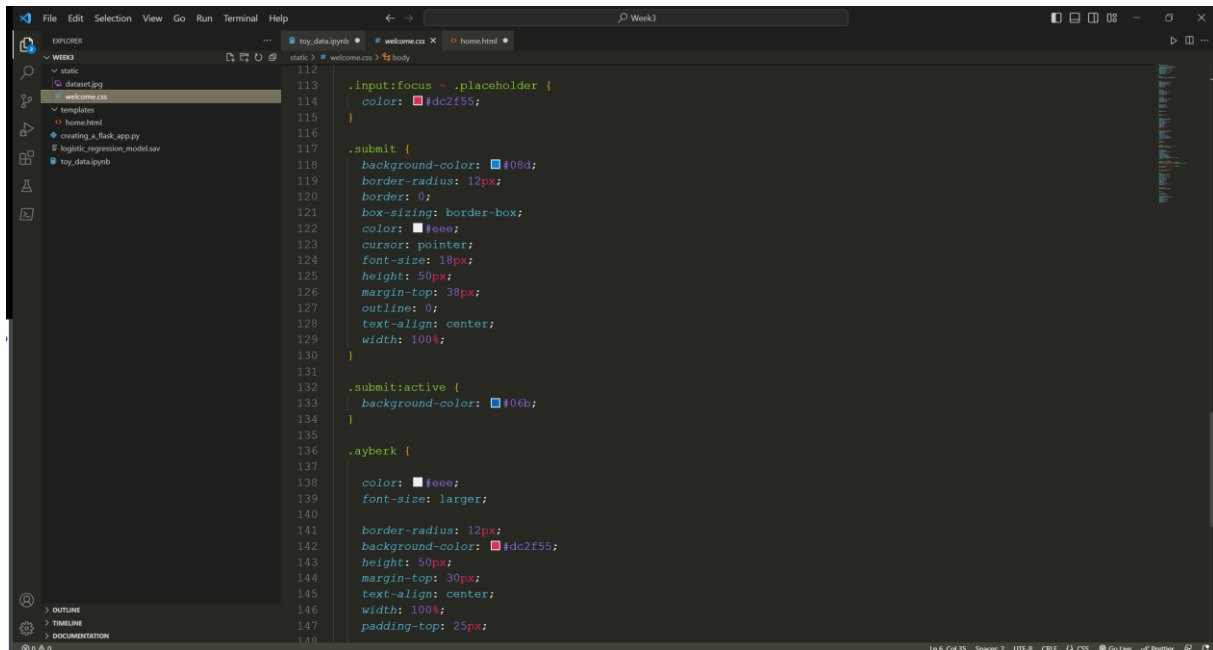
```
1 from flask import Flask, render_template, request
2
3 from flask import Flask, render_template, request
4 import pickle
5
6 model = pickle.load(open("logistic_regression_model.sav", "rb"))
7 target_names = list(
8     map(lambda x: x.title(), ('setosa', 'versicolor', 'virginica')))
9
10 app = Flask(__name__)
11
12
13 @app.route("/")
14 def home():
15     return render_template("home.html")
16
17
18 @app.route("/predict", methods=['POST'])
19 def predict():
20     features = [float(element) for element in request.form.values()]
21     prediction = int(model.predict([features]))
22     return render_template("home.html", prediction_text=target_names[prediction])
23
24
25 if __name__ == "__main__":
26     app.run(debug=True, port=5533)
```

4 – Creating an HTML for the front end.

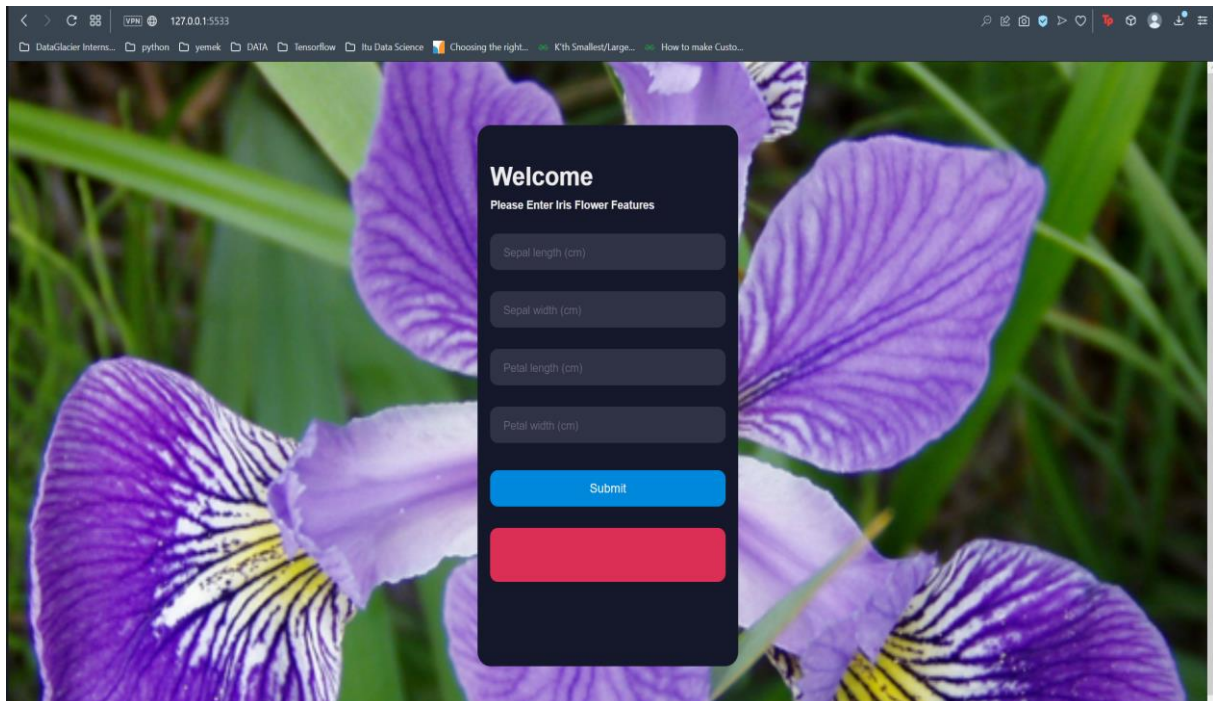


```
1 <link rel="stylesheet" href="/static/welcome.css">
2 <title>ML Deployment Trial</title>
3 </head>
4 <body>
5
6     <div class="form">
7         <form action="{{ url_for('predict') }}" method="post">
8             <div class="title">Welcome</div>
9             <div class="subtitle">Please Enter Iris Flower Features</div>
10             <div class="input-container ic2">
11                 <input id="Feaute1" class="input" type="text" placeholder=" " name="Feaute1"/>
12                 <div class="cut"></div>
13                 <label for="Feaute1" class="placeholder">Sepal length (cm)</label>
14             </div>
15             <div class="input-container ic2">
16                 <input id="Feaute2" class="input" type="text" placeholder=" " name="Feaute2"/>
17                 <div class="cut"></div>
18                 <label for="Feaute2" class="placeholder">Sepal width (cm)</label>
19             </div>
20             <div class="input-container ic2">
21                 <input id="Feaute3" class="input" type="text" placeholder=" " name="Feaute3"/>
22                 <div class="cut cut-short"></div>
23                 <label for="Feaute3" class="placeholder">Petal length (cm)</label>
24             </div>
25             <div class="input-container ic2">
26                 <input id="Feaute4" class="input" type="text" placeholder=" " name="Feaute4" />
27                 <div class="cut cut-short"></div>
28                 <label for="Feaute4" class="placeholder">Petal width (cm)</label>
29             </div>
30
31             <button type="submit" class="submit">Submit</button>
32
33         </form>
34         <div class="ayberk">{{ prediction_text }}</div>
35     </div>
```

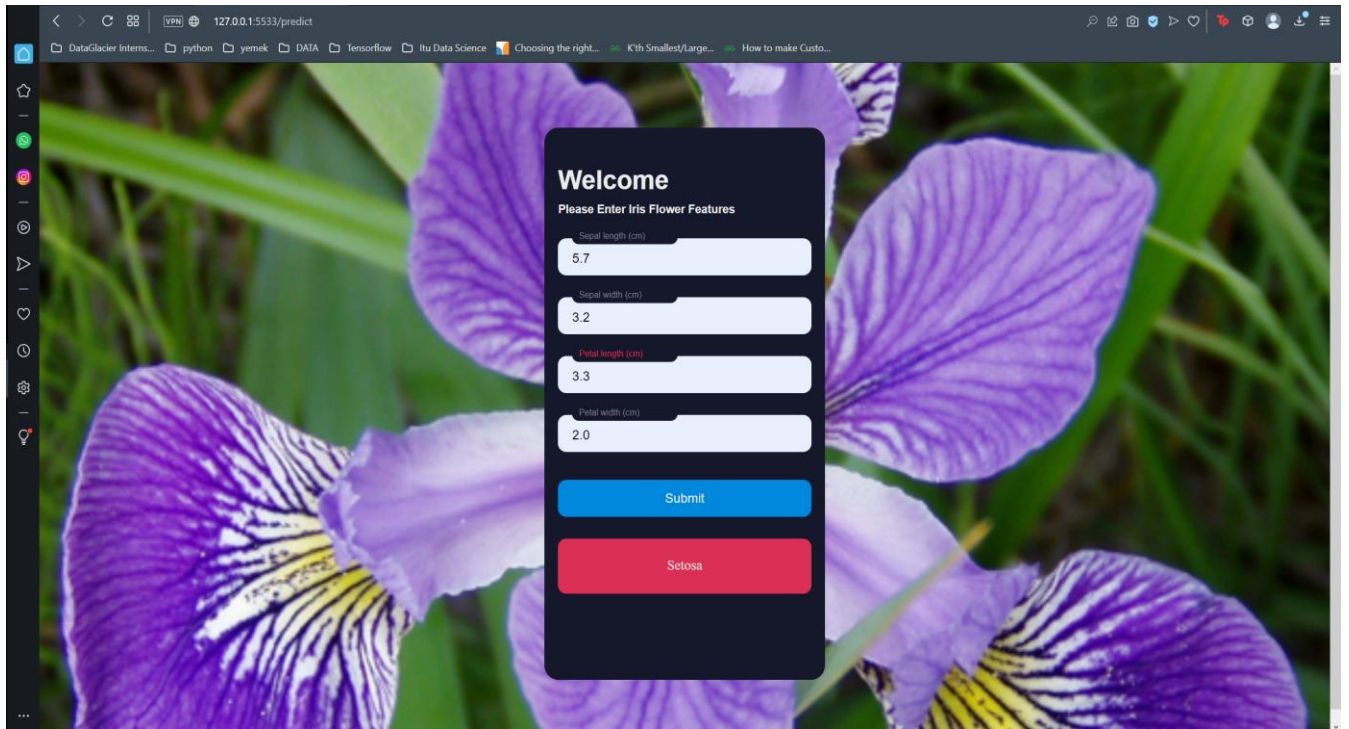
5 – Css for front end



6 – Result Website



7 – Testing the ML model on the Website



The screenshot shows a web browser window with a dark theme. The address bar displays '127.0.0.1:5533/predict'. The browser's tab bar shows several open tabs, including 'DataGlacier Intern...', 'python', 'yemek', 'DATA', 'Tensorflow', 'Itu Data Science', and three others with titles starting with 'Choosing the right...', 'K'th Smallest/Large...', and 'How to make Custo...'. The main content area features a background image of purple iris flowers. Overlaid on this is a dark gray modal form titled 'Welcome' with the subtitle 'Please Enter Iris Flower Features'. The form contains four input fields with labels and values: 'Sepal length (cm)' with '5.7', 'Sepal width (cm)' with '3.2', 'Petal length (cm)' with '3.3', and 'Petal width (cm)' with '2.0'. Below the inputs are two buttons: a blue 'Submit' button and a red 'Setosa' button. The left sidebar of the browser shows various icons for navigation and development tools.

Welcome

Please Enter Iris Flower Features

Sepal length (cm)
5.7

Sepal width (cm)
3.2

Petal length (cm)
3.3

Petal width (cm)
2.0

Submit

Setosa

It works!!