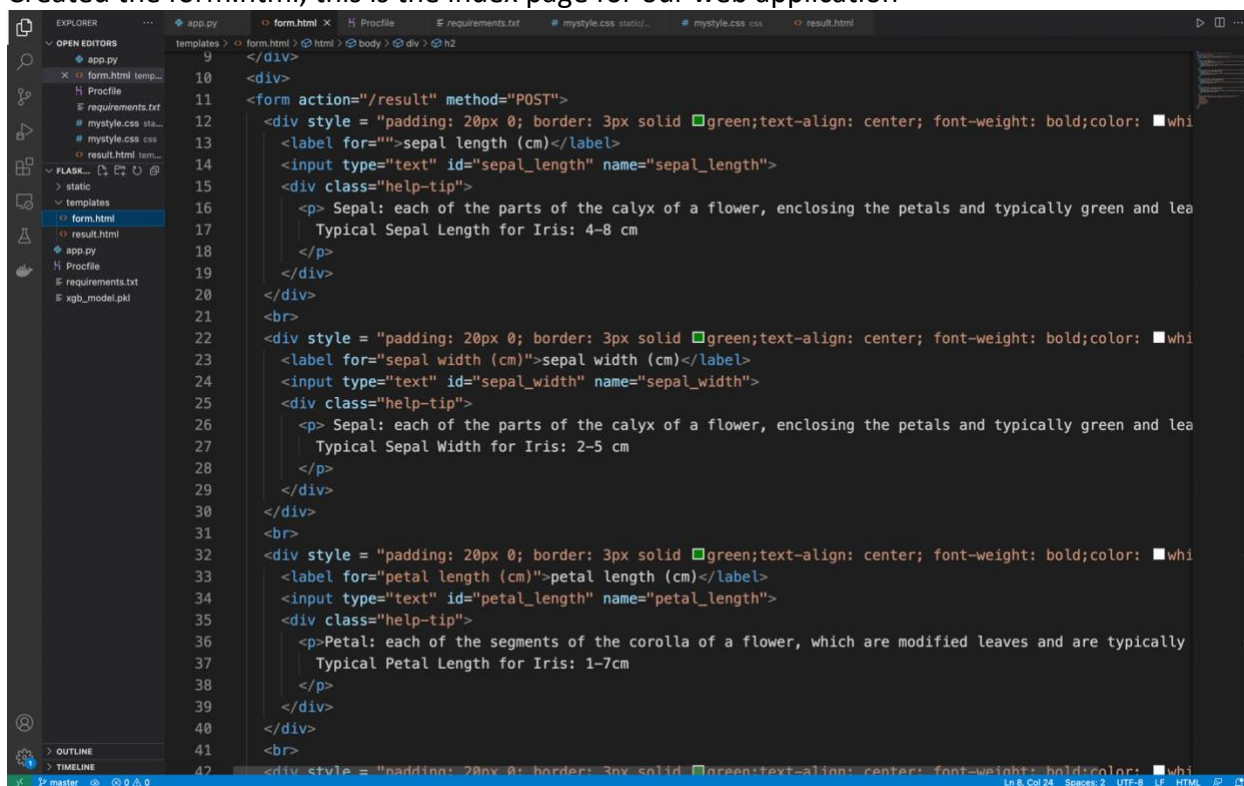
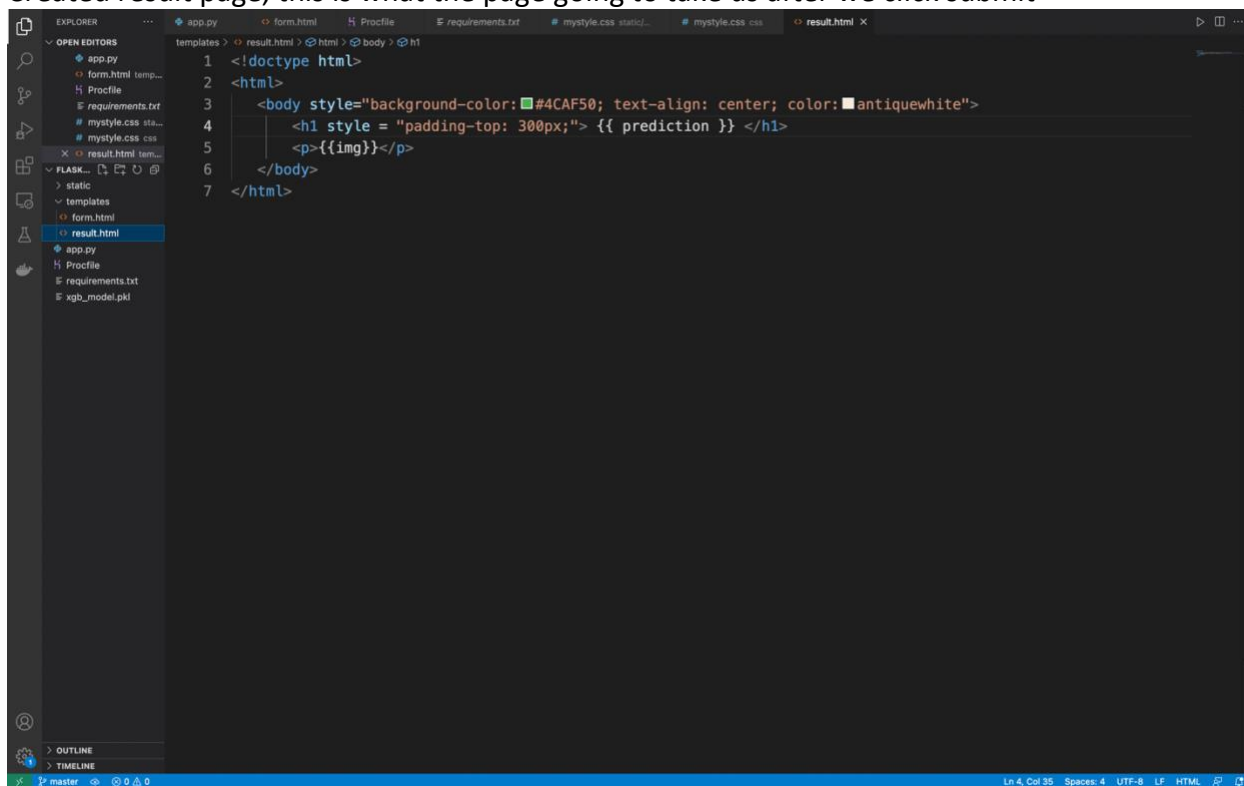


Created the form.html, this is the index page for our web application

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with files like app.py, form.html, and result.html. The main editor window displays the content of form.html. The code is an HTML form with three input fields for sepal length, sepal width, and petal length. Each input field is accompanied by a 'help-tip' div containing descriptive text about the measurement. The form uses inline styles for padding, border, text-align, font-weight, and color. The form action is set to '/result' and the method is 'POST'.

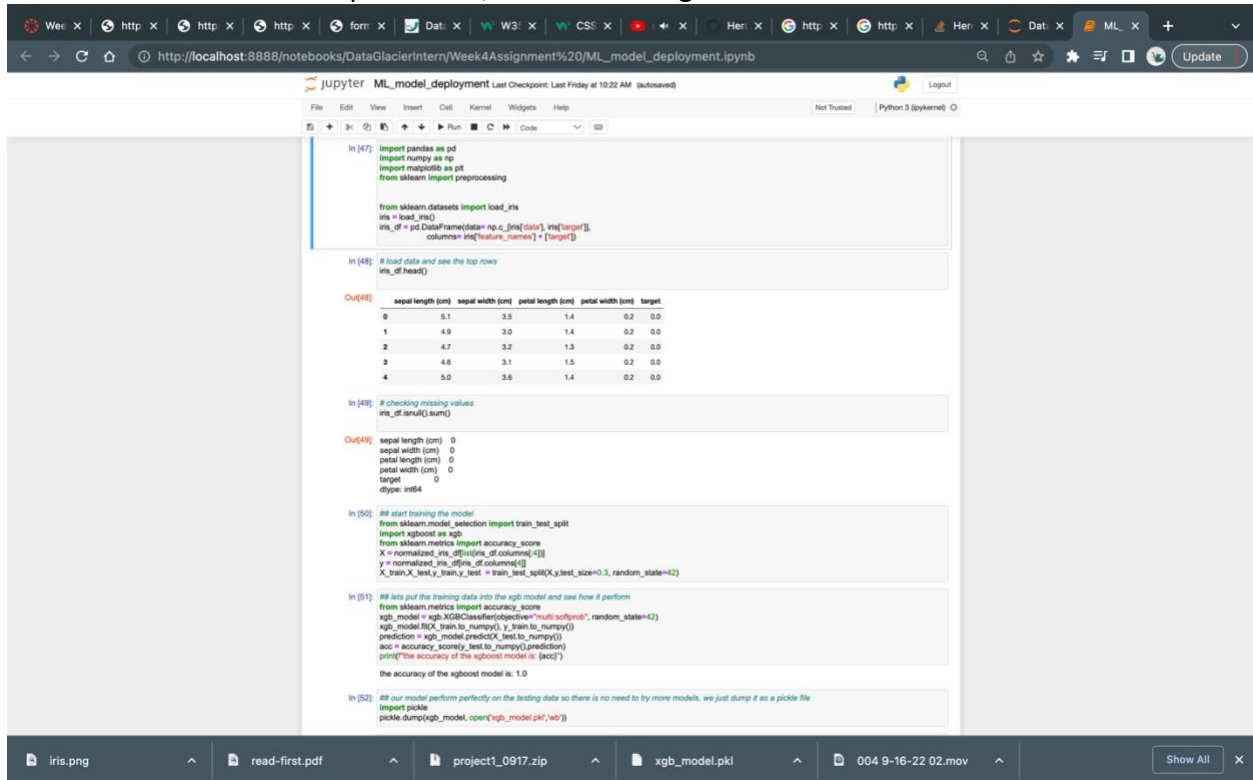
```
1 </div>
2
3 <div>
4   <form action="/result" method="POST">
5     <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white">
6       <label for="">sepal length (cm)</label>
7       <input type="text" id="sepal_length" name="sepal_length">
8       <div class="help-tip">
9         <p> Sepal: each of the parts of the calyx of a flower, enclosing the petals and typically green and lea
10         Typical Sepal Length for Iris: 4-8 cm
11       </p>
12     </div>
13   </div>
14   <br>
15   <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white">
16     <label for="sepal width (cm)">sepal width (cm)</label>
17     <input type="text" id="sepal_width" name="sepal_width">
18     <div class="help-tip">
19       <p> Sepal: each of the parts of the calyx of a flower, enclosing the petals and typically green and lea
20       Typical Sepal Width for Iris: 2-5 cm
21     </p>
22   </div>
23 </div>
24 <br>
25 <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white">
26   <label for="petal length (cm)">petal length (cm)</label>
27   <input type="text" id="petal_length" name="petal_length">
28   <div class="help-tip">
29     <p>Petal: each of the segments of the corolla of a flower, which are modified leaves and are typically
30     Typical Petal Length for Iris: 1-7cm
31   </p>
32 </div>
33 </div>
34 <br>
35 <div style = "padding: 20px 0; border: 3px solid green;text-align: center; font-weight: bold;color: white">
36   <div style="padding-top: 300px;"> {{ prediction }} </div>
37 </div>
```

Created result page, this is what the page going to take us after we click submit

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows the project structure. The main editor window displays the content of result.html. The code is a simple HTML page with a body background color of #4CAF50 and text-align center. It contains a single line of code for a prediction, which is currently empty: {{ prediction }}.

```
1 <!doctype html>
2 <html>
3   <body style="background-color: #4CAF50; text-align: center; color: antiquewhite">
4     <h1 style = "padding-top: 300px;"> {{ prediction }} </h1>
5     <p>{{img}}</p>
6   </body>
7 </html>
```

Created the model for Iris prediction, the model is xgboost.



```
In [47]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing

from sklearn.datasets import load_iris
iris = load_iris()
iris_df = pd.DataFrame(data=np.c_[iris['data'], iris['target']],
    columns=iris['feature_names'] + ['target'])

In [48]: # load data and see the top rows
iris_df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

```
In [49]: # checking missing values
iris_df.isnull().sum()

Out[49]: sepal length (cm)    0
sepal width (cm)           0
petal length (cm)          0
petal width (cm)           0
target                     0
dtype: int64

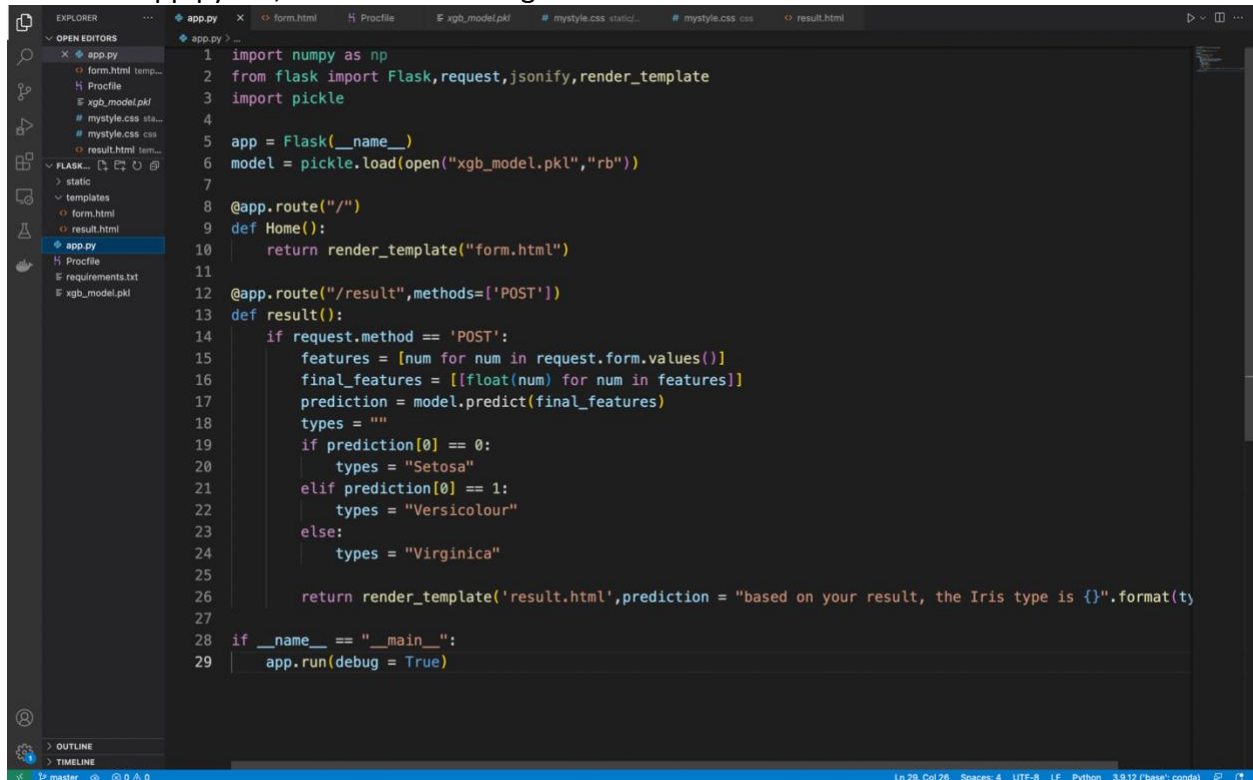
In [50]: ## start training the model
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import accuracy_score
X = normalized_iris_df[iris_df.columns[0:4]]
y = normalized_iris_df[iris_df.columns[5]]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

In [51]: ## lets put the training data into the xgb model and see how it perform
from sklearn.metrics import accuracy_score
xgb_model = xgb.XGBClassifier(objective='multi:softmax', random_state=42)
xgb_model.fit(X_train.to_numpy(), y_train.to_numpy())
prediction = xgb_model.predict(X_test.to_numpy())
acc = accuracy_score(y_test.to_numpy(), prediction)
print("the accuracy of the xgboost model is: {acc}")

the accuracy of the xgboost model is: 1.0

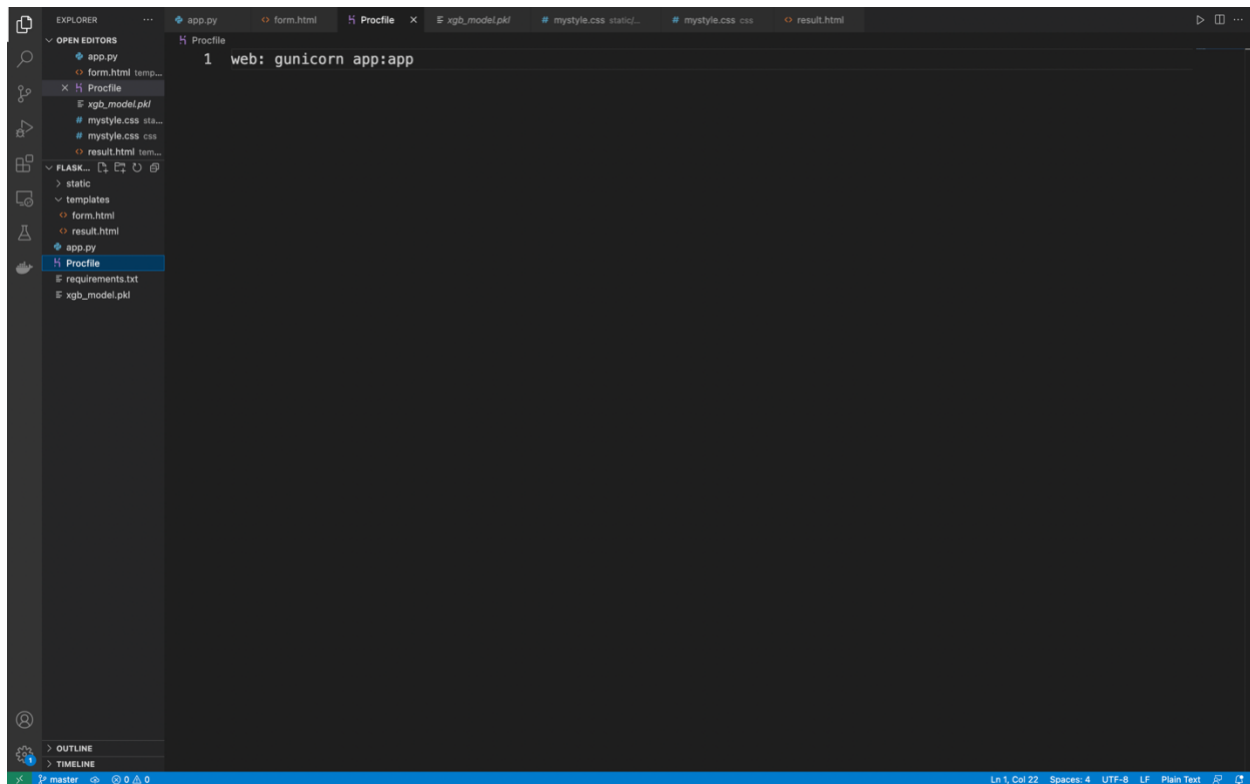
In [52]: ## our model perform perfectly on the testing data so there is no need to try more models, we just dump it as a pickle file
import pickle
pickle.dump(xgb_model, open("xgb_model.pkl", "wb"))
```

Write the app.py file, it is a written using flask



```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open("xgb_model.pkl", "rb"))
7
8 @app.route("/")
9 def Home():
10     return render_template("form.html")
11
12 @app.route("/result", methods=['POST'])
13 def result():
14     if request.method == 'POST':
15         features = [num for num in request.form.values()]
16         final_features = [[float(num) for num in features]]
17         prediction = model.predict(final_features)
18         types = ""
19         if prediction[0] == 0:
20             types = "Setosa"
21         elif prediction[0] == 1:
22             types = "Versicolour"
23         else:
24             types = "Virginica"
25
26         return render_template('result.html', prediction = "based on your result, the Iris type is {}".format(types))
27
28 if __name__ == "__main__":
29     app.run(debug = True)
```

Created Procfile for heroku



The screenshot shows the Visual Studio Code editor with a file explorer on the left. The 'Procfile' file is open in the editor, containing the following text:

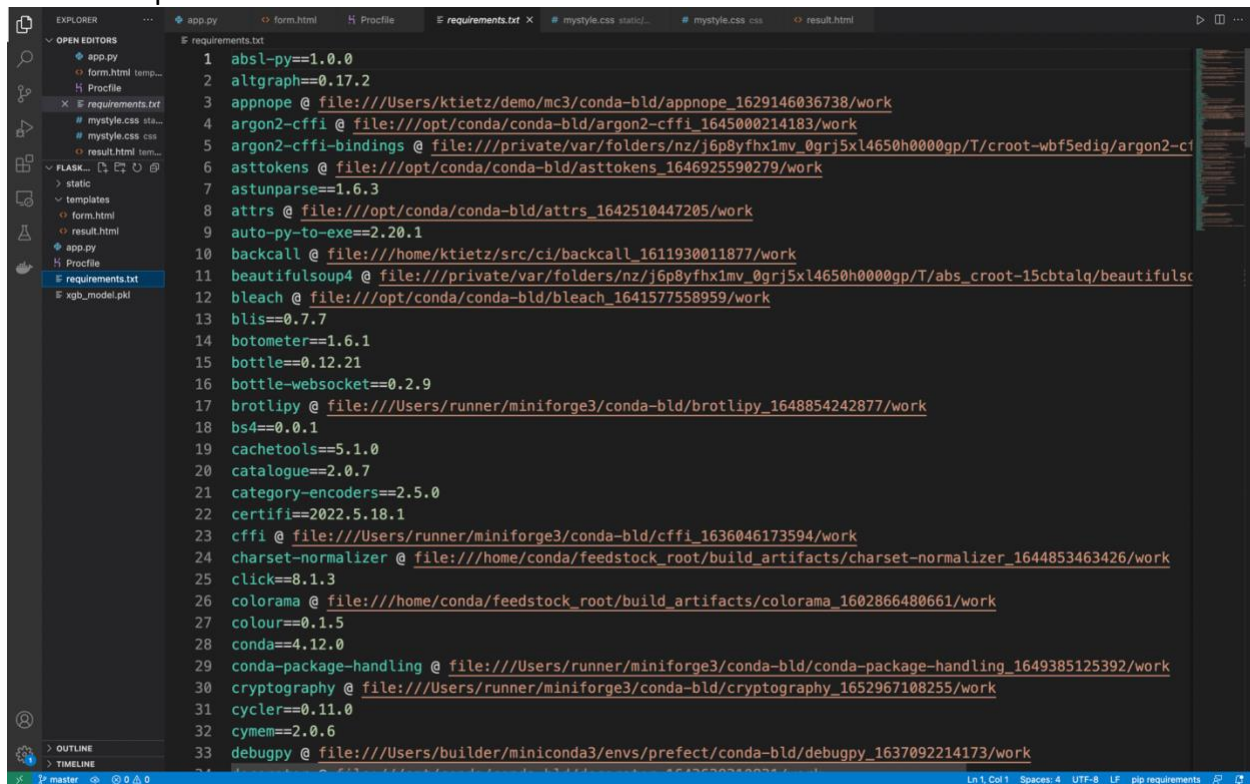
```
1 web: gunicorn app:app
```

The file explorer shows the following files and folders:

- app.py
- form.html
- Procfile
- xgb_model.pkl
- mystyle.css
- static
- templates
- form.html
- result.html
- app.py
- Procfile
- requirements.txt
- xgb_model.pkl

The status bar at the bottom indicates the file is at line 1, column 22, with 4 spaces, in UTF-8 encoding, LF line endings, and Plain Text format.

Created requirements.txt



The screenshot shows the Visual Studio Code editor with a file explorer on the left. The 'requirements.txt' file is open in the editor, containing the following text:

```
1 absl-py==1.0.0
2 altgraph==0.17.2
3 appnope @ file:///Users/ktietz/demo/mc3/conda-bld/appnope_1629146036738/work
4 argon2-cffi @ file:///opt/conda/conda-bld/argon2-cffi_164500214183/work
5 argon2-cffi-bindings @ file:///private/var/folders/nz/j6p8yfhx1mv_0grj5x14650h0000gp/T/croot-wbf5edig/argon2-cffi_164500214183/work
6 asttokens @ file:///opt/conda/conda-bld/asttokens_1646925590279/work
7 astunparse==1.6.3
8 attrs @ file:///opt/conda/conda-bld/attrs_1642510447205/work
9 auto-py-to-exe==2.20.1
10 backcall @ file:///home/ktietz/src/ci/backcall_1611930011877/work
11 beautifulsoup4 @ file:///private/var/folders/nz/j6p8yfhx1mv_0grj5x14650h0000gp/T/abs_croot-15cbtalq/beautifulsoup4_1644853463426/work
12 bleach @ file:///opt/conda/conda-bld/bleach_1641577558959/work
13 blis==0.7.7
14 botometer==1.6.1
15 bottle==0.12.21
16 bottle-websocket==0.2.9
17 brotli @ file:///Users/runner/miniforge3/conda-bld/brotli_1648854242877/work
18 bs4==0.0.1
19 cachetools==5.1.0
20 catalogue==2.0.7
21 category-encoders==2.5.0
22 certifi==2022.5.18.1
23 cffi @ file:///Users/runner/miniforge3/conda-bld/cffi_1636046173594/work
24 charset-normalizer @ file:///home/conda/feedstock_root/build_artifacts/charset-normalizer_1644853463426/work
25 click==8.1.3
26 colorama @ file:///home/conda/feedstock_root/build_artifacts/colorama_1602866480661/work
27 colour==0.1.5
28 conda==4.12.0
29 conda-package-handling @ file:///Users/runner/miniforge3/conda-bld/conda-package-handling_1649385125392/work
30 cryptography @ file:///Users/runner/miniforge3/conda-bld/cryptography_1652967108255/work
31 cycycler==0.11.0
32 cymem==2.0.6
33 debugpy @ file:///Users/builder/miniconda3/envs/prefect/conda-bld/debugpy_1637092214173/work
```

The file explorer shows the following files and folders:




- app.py
- form.html
- Procfile
- requirements.txt
- mystyle.css
- static
- templates
- form.html
- result.html
- app.py
- Procfile
- requirements.txt
- xgb_model.pkl

The status bar at the bottom indicates the file is at line 1, column 1, with 4 spaces, in UTF-8 encoding, LF line endings, and pip requirements format.

Make sure our webapp run perfectly

Week 5: x Data Sc: x W3Sch: x CSS for: x (1) x Heroku: x https:// x https:// x Heroku: x DataG: x iris-pre: x https:// x +

← → ↻ 🏠 https://iris-predictor-01.herokuapp.com ⚙️ ☆ ⚙️ 📄 🔄 Update



Iris Versicolor Iris Setosa Iris Virginica

Iris Classifier

sepal length (cm) ?

sepal width (cm) ?

petal length (cm) ?

petal width (cm) ?

Week 5: x Data Sc: x W3Sch: x CSS for: x (1) x Heroku: x https:// x https:// x Heroku: x DataG: x iris-pre: x https:// x +

← → ↻ 🏠 https://iris-predictor-01.herokuapp.com/result ⚙️ ☆ ⚙️ 📄 🔄 Update

based on your result, the Iris type is Virginica