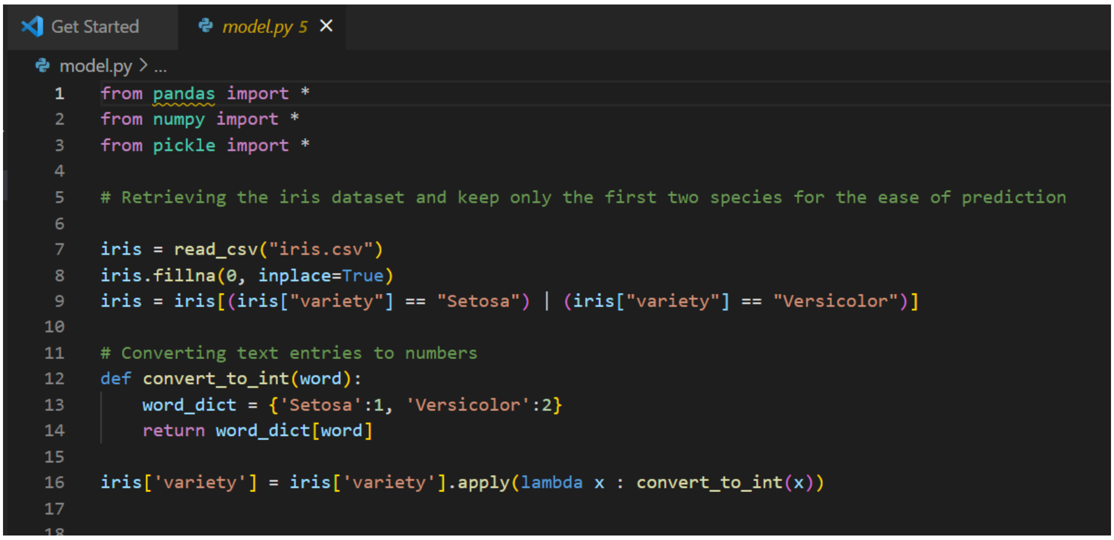
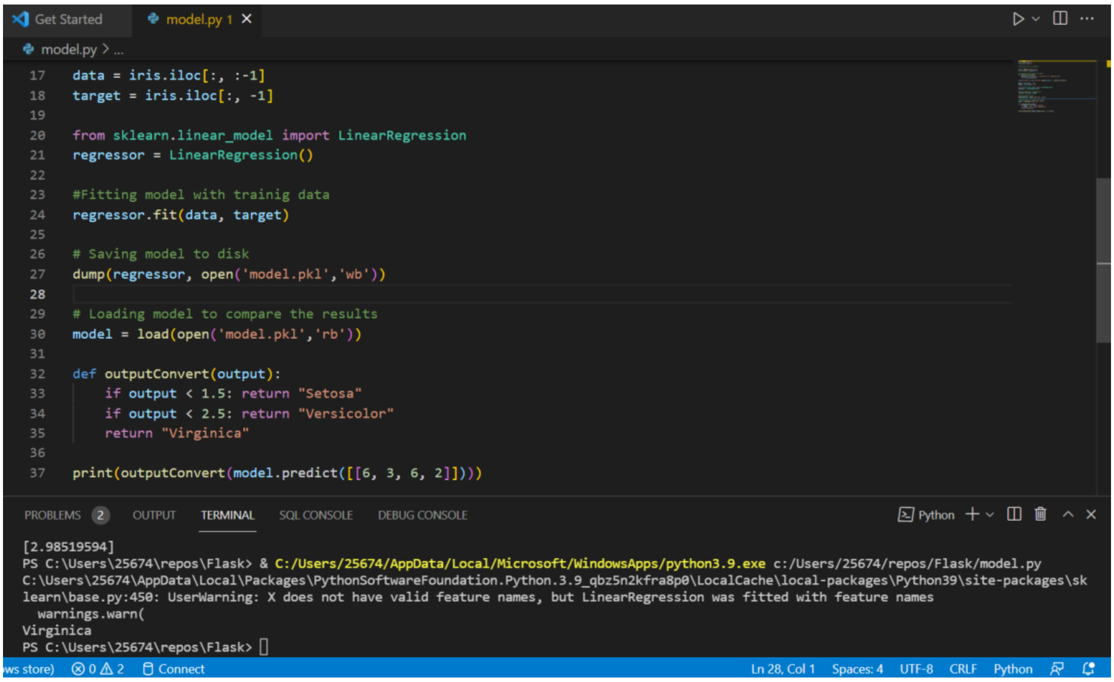


```python
from pandas import *
from numpy import *
from pickle import *

# Retrieving the iris dataset and keep only the first two species for the ease of prediction

iris = read_csv("iris.csv")
iris.fillna(0, inplace=True)
iris = iris[(iris["variety"] == "Setosa") | (iris["variety"] == "Versicolor")]

# Converting text entries to numbers
def convert_to_int(word):
    word_dict = {'Setosa':1, 'Versicolor':2}
    return word_dict[word]

iris['variety'] = iris['variety'].apply(lambda x : convert_to_int(x))

```

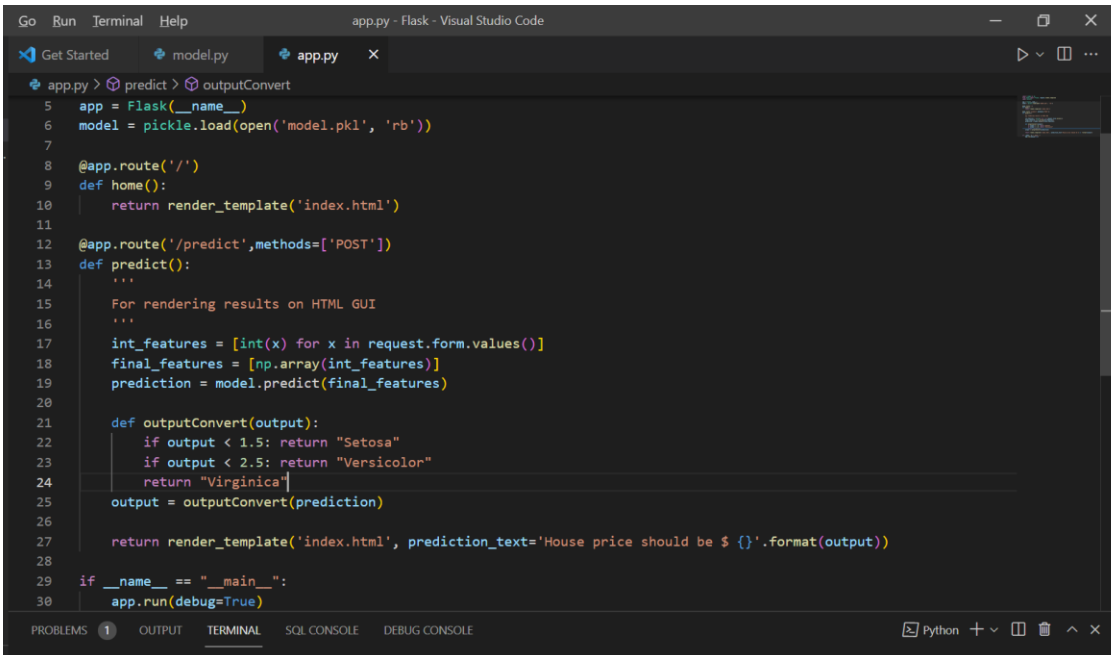
Checkpoint: retrieving the iris dataset from scikit-learn



```python
data = iris.iloc[:, :-1]
target = iris.iloc[:, -1]

from sklearn.linear_model import LinearRegression
regressor = LinearRegression()

#Fitting model with trainig data
regressor.fit(data, target)

# Saving model to disk
dump(regressor, open('model.pkl','wb'))

# Loading model to compare the results
model = load(open('model.pkl','rb'))

def outputConvert(output):
    if output < 1.5: return "Setosa"
    if output < 2.5: return "Versicolor"
    return "Virginica"

print(outputConvert(model.predict([[6, 3, 6, 2]])))
```

```
[2.98519594]
PS C:\Users\25674\repos\Flask> & C:/Users/25674/AppData/Local/Microsoft/WindowsApps/python3.9.exe c:/Users/25674/repos/Flask/model.py
C:\Users\25674\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.9_qbz5n2kfra8p0\LocalCache\local-packages\Python39\site-packages\sk
learn\base.py:450: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
Virginica
PS C:\Users\25674\repos\Flask> 
```

Checkpoint: modify the model part to match our iris dataset and make a test prediction



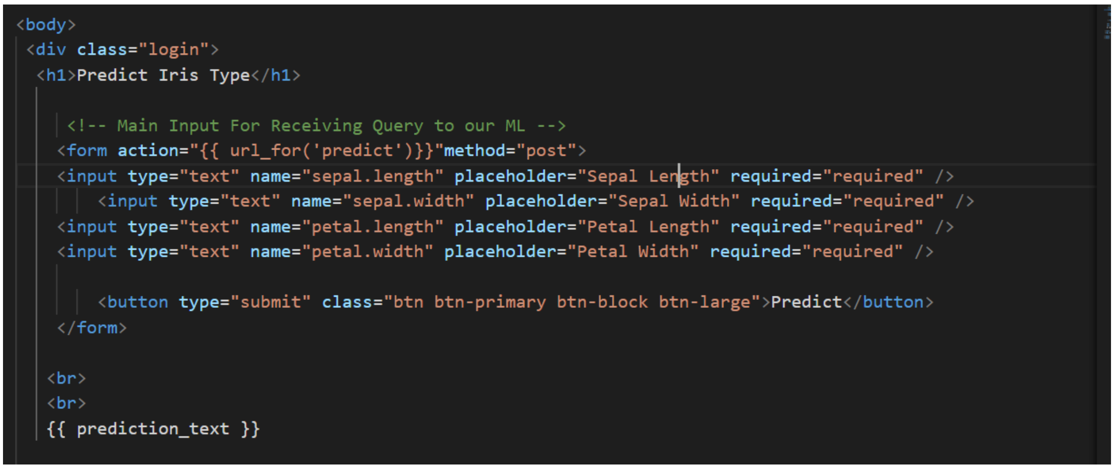
```
5   app = Flask(__name__)
6   model = pickle.load(open('model.pkl', 'rb'))
7
8   @app.route('/')
9   def home():
10      return render_template('index.html')
11
12  @app.route('/predict',methods=['POST'])
13  def predict():
14      '''
15      For rendering results on HTML GUI
16      '''
17      int_features = [int(x) for x in request.form.values()]
18      final_features = [np.array(int_features)]
19      prediction = model.predict(final_features)
20
21      def outputConvert(output):
22          if output < 1.5: return "Setosa"
23          if output < 2.5: return "Versicolor"
24          return "Virginica"
25      output = outputConvert(prediction)
26
27      return render_template('index.html', prediction_text='House price should be $ {}'.format(output))
28
29  if __name__ == "__main__":
30      app.run(debug=True)
```

Checkpoint: slightly modify app.py to match our model

** Modification: change type “int” to “float”

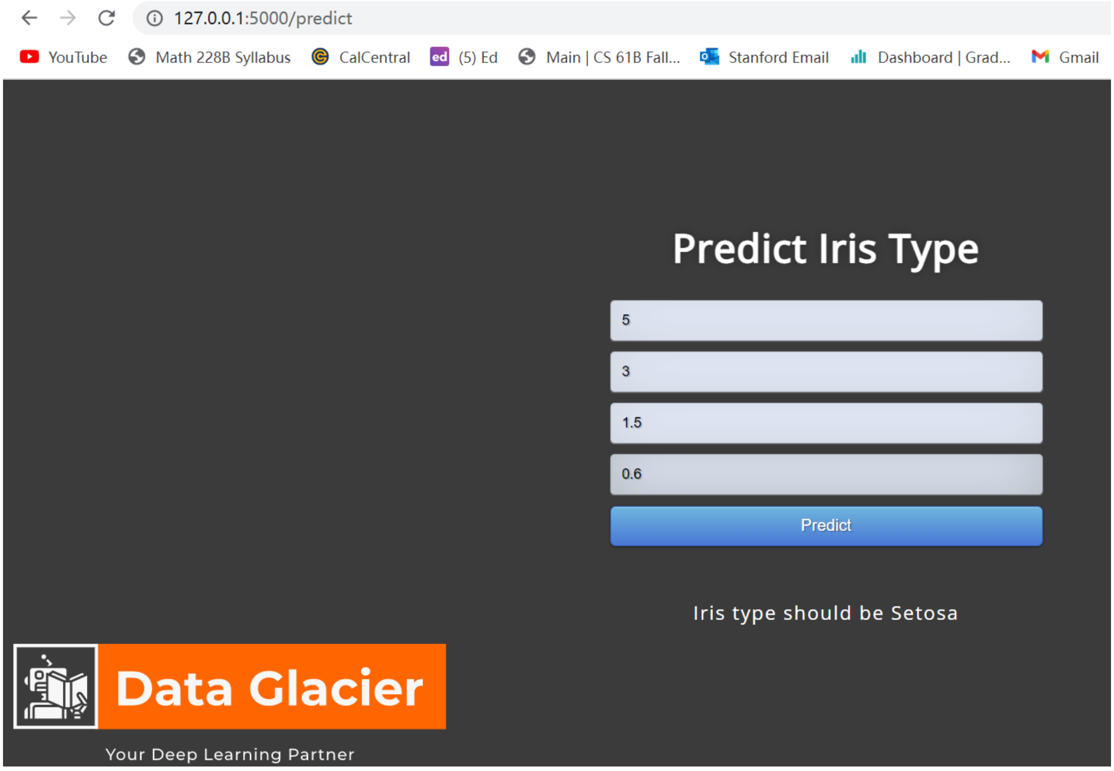
** Modification: remove the dollar sign in line 27

** Modification: change “House type” to “Iris type” in line 27.



```
<body>
 <div class="login">
  <h1>Predict Iris Type</h1>

    <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('predict')}}"method="post">
    <input type="text" name="sepal.length" placeholder="Sepal Length" 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" />

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
    </form>

   <br>
   <br>
   {{ prediction_text }}
```

Checkpoint: slightly modify index.html to match our inputs



Predict Iris Type

5

3

1.5

0.6

Iris type should be Setosa

Final snapshot of our predictor webpage