Loading the Iris dataset and creating the XGBoosting Model, then save the model into a pickle file

In [47]:
```
import pandas as pd
import numpy as np
import matplotlib 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()
```

Out[48]:

	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[list(iris_df.columns[:4])]
y = normalized_iris_df[iris_df.columns[4]]
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:softprob", 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(f"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'))
```

Creating the index.html (landing page)

```
<html>
<body>
    <h3>Iris Classification App</h3>

<div>
  <form action="/result" method="POST">
    <label for="">sepal length (cm)</label>
    <input type="text" id="sepal_length" name="sepal_length">
    <br>
    <label for="sepal width (cm)">sepal width (cm)</label>
    <input type="text" id="sepal_width" name="sepal_width">
    <br>
    <label for="petal length (cm)">petal length (cm)</label>
    <input type="text" id="petal_length" name="petal_length">
    <br>
    <label for="petal width (cm)">petal width (cm)</label>
    <input type="text" id="petal_width" name="petal_width">
    <br>
    <input type="submit" value="Submit">
  </form>
</div>
</body>
</html>
```

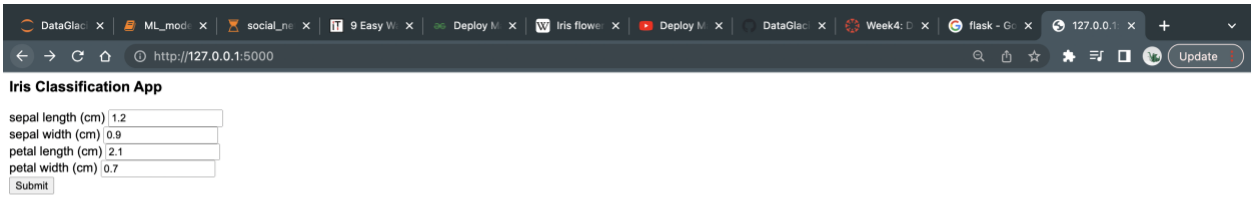
Creating result page

```
<!doctype html>
<html>
    <body>
        <h1> {{ prediction }} </h1>
    </body>
</html>
```

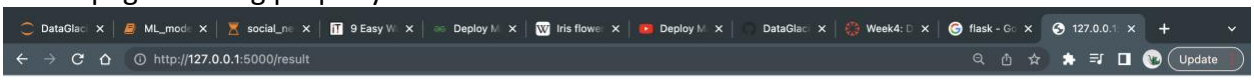
Creating app.py using flask framework

```python
from flask import Flask,request,jsonify,render_template
import pickle

app = Flask(__name__)
model = pickle.load(open("xgb_model.pkl","rb"))

@app.route("/")
def Home():
    return render_template("form.html")

@app.route("/result",methods=['POST'])
def result():
    if request.method == 'POST':
        features = [num for num in request.form.values()]
        final_features = [[float(num) for num in features]]
        prediction = model.predict(final_features)
        types = ""
        if prediction[0] == 0:
            types = "Setosa"
        elif prediction[0] == 1:
            types = "Versicolour"
        else:
            types = "Virginica"

        return render_template('result.html',prediction = "based on your result, the Iris type is {}".format(ty
    __name__: str
if __name__ == "__main__":
    app.run(debug = True)
```

Index page working properly



Result page working properly



based on your result, the Iris type is Virginica