Cross validation with deploymen	
	from sklearn.datasets import load_iris from sklearn.linear_model import LogisticRegression from sklearn.model_selection import KFold, cross_val_score #We'll also use the cross_val_score function to evaluate our model across the k folds #For demonstration purposes, we'll use the Iris dataset and a simple classifier (LogisticRegression).
	#Load the dataset: data = load_iris() X, y = data.data, data.target
	data.target # Set up k-fold cross-validation:
	#We'll use 5-fold cross-validation in this example. k = 5 kf = KFold(n_splits=k, shuffle=True, random_state=43) #Random state here because it want wheneve you run it gives same data points in that 5 box #n_splits=k specifies the number of folds. #shuffle=True ensures that the data is shuffled before splitting into folds. #random_state=42 is an arbitrary seed for reproducibility.
	#Apply k-fold cross-validation using a classifier:
	#We'll use the cross_val_score function to evaluate the performance of our LogisticRegression classifier across the k folds. clf = LogisticRegression(max_iter=200, random_state=42)
	<pre>##max_iter=200: This specifies the maximum number of iterations the solver (i.e., the algorithm used to find the logistic regression coefficients) will take to converge. The default value is 100, but sometimes, increasing this number can help in cases where the solver might not converge quickly. #Max_iter- This parameter is used when you call the fit method on a logistic regression model to train it on a specific dataset.</pre>
	<pre>#Performing Cross-Validation: scores = cross_val_score(clf, X, y, cv=kf)</pre>
	#cv=kf: This specifies the cross-validation splitting strategy to be used. In our case, we're using the kf object we created earlier, which is a 5-fold cross-validation strategy with shuffling.

```
#scores will contain the accuracy of the classifier for each of the
k folds.
scores #1. means 100% accuracy
#Analyze the results:
#We can compute the average and standard deviation of the scores to
get an idea of the model's performance and variability across the
folds.
average_accuracy = scores.mean()
std accuracy = scores.std()
#Approximately 97.33%
average accuracy
#Standard Deviation of Accuracy: Approximately 2.49% ( it means
2.49% var from mean of all accuracy)
std_accuracy
#Note
#1. Cross validation do fit the model and discard- the purpose of this not to train model
and use for prediction - the purpose of this to find accuray of differene - new data
#2.If you want to train model and use for prediction then you need to fit again and do
prediction
#Supose if you want to train model for prediction
clf = LogisticRegression(max iter=200, random state=42)
clf.fit(X, y)
#Now, you can use this trained model (clf) to make predictions on new data. Let's say
you have some random data new_data:
new data = [[5.0, 3.5, 1.5, 0.2]]
predicted class = clf.predict(new data)
predicted class value = predicted class[0]
predicted class value
#0 means setosa
#Install flask ngrok, to expose this enviroment publicly
!pip install pyngrok
```

```
from flask import Flask, request, jsonify
from pyngrok import ngrok
import numpy as np
import joblib
# Set your Ngrok authentication token
ngrok.set auth token('2agBTjLoJNULDJEaUNJD5hWnNkc 4hi26fzaejcQ3UjQhWaNB')
app = Flask( name )
# Load pre-trained model (replace 'your model.pkl' with the actual model file)
model = clf
@app.route('/predict', methods=['POST'])
def predict():
  try:
     # Mapping of class labels to flower names
     class_names = {0: "setosa", 1: "versicolor", 2: "virginica"}
     data = request.json
     prediction = model.predict(np.array([data['input']]).reshape(1, -1))
     flower_name = class_names[int(prediction[0])]
     return jsonify({'prediction': int(prediction[0]), 'flower_name': flower_name})
  except Exception as e:
     return jsonify({'error': str(e)})
if __name__ == '__main__':
  # Use pyngrok to create a tunnel and run the Flask app
  ngrok tunnel = ngrok.connect(5000)
  print(" * ngrok tunnel \"{}\" ->
\"http://127.0.0.1:{}/\"".format(ngrok_tunnel.public_url, 5000))
  app.run()
#To stop server
ngrok.kill()
            except Exception as e:
               return jsanify(['error': str(e)))
           __name _ - _ _min__':

# Doe pyngrok to create a tunnel and row the (lack app
ngrok tunnel - ngrok.connect(5808)
print(" * egrok tunnel \"()\" -> \"http://137.e.e.l:{}/\"- format(ngrok_tunnel.public_url, 5008))
   * ngrok tunnel "https://5f3m-14-74-288-19.ngrok-free.amp" -> "http://127.8.8.115888//"
* Serving Flask app '__main__'
* Debug mode: off
        IMFO:workzoug:walNING: This is a development server. Do not use it in a production deployment. Use a production MSGI
* Running on http://lif.u.o.ii.5000
       IMFO:werkZeug:Press CTRL+C to quit
IMFO:werkzeug:127.0.0.1 - [12/Jan/2024 85:54:45] "POST / HTTP/1.1" 404 -
IMFO:werkzeug:127.0.0.1 - [12/Jan/2024 85:55:87] "POST /predict HTTP/1.1" 200
```

```
save the
model and
deploy it
                import joblib
                your_model = clf
               filename = "mymodel deployment"
                joblib.dump(clf,filename)
                print(f'Model saved to {filename}')
File is
                                                                + Code + Text
                                                                                    ♠ Copy to Drive
                                                     □ ×
                 ⊞ Files
Created
                                                                        File "/usr/local/lib/python3.10/dist-packages/requests/api
                                                                        return request("get", url, params-params, "kwargs)
File "/usr/local/lib/python3.10/dist-packages/requests/api.
return session.request(method=method, url=url, "*kwargs)
File "/usr/local/lib/python3.10/dist-packages/requests/sess
                        a
                 \{x\}
                                                                        resp = self.send(prep, "send_kwargs)

File "/usr/local/lib/pythona.l0/dist-packages/requests/sess
r = adapter.send(request, "kwargs)

File "/usr/local/lib/pythona.l0/dist-packages/requests/adap

    Ite sample_data

                         mymodel_deployment
                                                                       raise ConnectionError(e, request-request)
requests.exceptions.ConnectionError: HTTPConnectionPool(host
                 import joblib
                                                                       your model - clf
                                                                       filename = "mymodel_deployment"
                                                                       joblib.dump(clf,filename)
                                                                       print(f'Model saved to (filename)')

→ Model saved to mymodel_deployment

First
Upload the
File in New # prompt: No module named 'pyngrok'
Machine
                !pip install pyngrok
and Run
the Code; from flask import Flask, request, isonify
Before that from pyngrok import ngrok
                import numpy as np
add the
AuthToken import joblib
in Code
and file
name
               # Set your Ngrok authentication token
                ngrok.set_auth_token('2aqNEGImK2gUAsDcf2qQNLFV3qS_6ETQCYPCXv8PC6cxM7GnX')
               app = Flask(__name___)
               # Load pre-trained model (replace 'your_model.pkl' with the actual model file)
               filename = 'mymodel deployment'
               # Load the model from the file
                model = joblib.load(filename)
```

```
@app.route('/predict', methods=['POST'])
               def predict():
                 try:
                    # Mapping of class labels to flower names
                    class_names = {0: "setosa", 1: "versicolor", 2: "virginica"}
                    data = request.json
                    prediction = model.predict(np.array([data['input']]).reshape(1, -1))
                    flower_name = class_names[int(prediction[0])]
                    return jsonify({'prediction': int(prediction[0]), 'flower name': flower name})
                 except Exception as e:
                    return jsonify({'error': str(e)})
               if __name__ == '__main__':
                 # Use pyngrok to create a tunnel and run the Flask app
                 ngrok_tunnel = ngrok.connect(5000)
                 print(" * ngrok tunnel \"{}\" ->
               \"http://127.0.0.1:{}/\"".format(ngrok_tunnel.public_url, 5000))
                 app.run()
                " ngrok tunnel "https://A659-15-194-136-97.ngrok-free.app" > "http://127.0.0.1:5008/"
" Serving Flask app '__main_'
Copy the
ngrok
                      * Debug mode: off
                     INFO:werkzeug:MARNING: This is a development server. Do not use it in a production deployment. Use a production * Running on http://izz.e.e.iiiide
tunnal URL
and past in
                                              [12/3an/2024 06:50:07] "POST /predict HTTP/1.1" 200
HTTP
request
Select
                Home Workspaces - API Network -
                                                                       Q Search Postman
                                                                                                                  A 0
Request
                                                                           PROF NUMBE //4859-35-194-1 • +
type as
                 Save -
                                                          frttps://4659-35-194-136-97.ngrox-free.app/predict
HTTP and
past ngrok
                                                                       https://4659-35-194-136-97.ngrok-free.applipredict
                 POST
tunnal URL
                                                         Perame Auth Headurs (7) Body • Pre-req. Tests Settings
                 0
In Text
                                                                 JSON -
Field,
                 00
select
                                                                  'input': [5.3, 3.5, 1.4, 0.2]
perameter
style as
                             Create a collection for your
                                                                                                    🖺 200 OK 1120 ms 237 8 🖾 Sa
                                   requests
raw and in
                                                         Protty
                                                                 New Preview Visualize Json v
body
                              requests and easily set common
                                                                  "flower_name": "setose",
"prediction": 0
                              authorization, tests, ecripts, and
Add input
                              variables for all requests in it.
Load the
model
Created
               Db[]
empty
variable
```

Create two api's i.e predict & update	
Store the data in model(train)	
Predict	
Update(if it predict wrong)	
Check accuracy	