Deployment

What is Model Deployment?

Deployment is the method by which you integrate a machine learning model into an existing production environment to make practical business decisions based on data. It is one of the last stages in the machine learning life cycle and can be one of the most cumbersome. Often, an organization's IT systems are incompatible with traditional model-building languages, forcing data scientists and programmers to spend valuable time and brainpower rewriting them.

Why is Model Deployment Important?

In order to start using a model for practical decision-making, it needs to be effectively deployed into production. If you cannot reliably get practical insights from your model, then the impact of the model is severely limited.

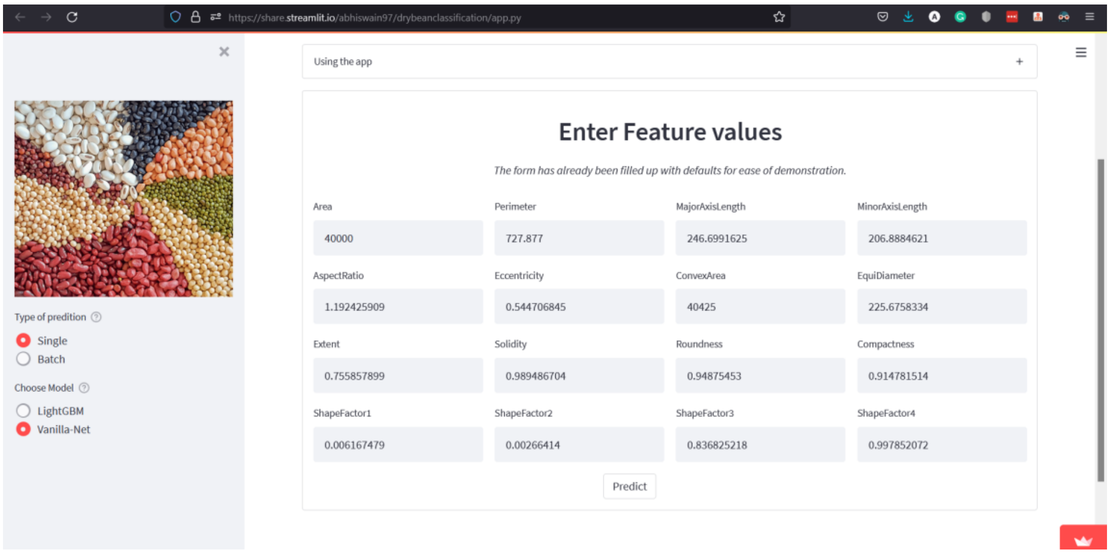
Model deployment is one of the most difficult processes of gaining value from machine learning. It requires coordination between data scientists, IT teams, software developers, and business professionals to ensure the model works reliably in the organization's production environment. This presents a major challenge because there is often a discrepancy between the programming language in which a machine learning model is written and the languages your production system can understand, and re-coding the model can extend the project timeline by weeks or months.

In order to get the most value out of machine learning models, it is important to seamlessly deploy them into production so a business can start using them to make practical decisions.

Deploying our app

I have created a Streamlit app named “Dry bean classifier”. It has two ways you can make classification.

1. Single prediction



https://share.streamlit.io/abhiswain97/drybeanclassification/app.py

Type of predition

● Single
○ Batch

Choose Model

○ LightGBM
● Vanilla-Net

Using the app +

Enter Feature values

The form has already been filled up with defaults for ease of demonstration.

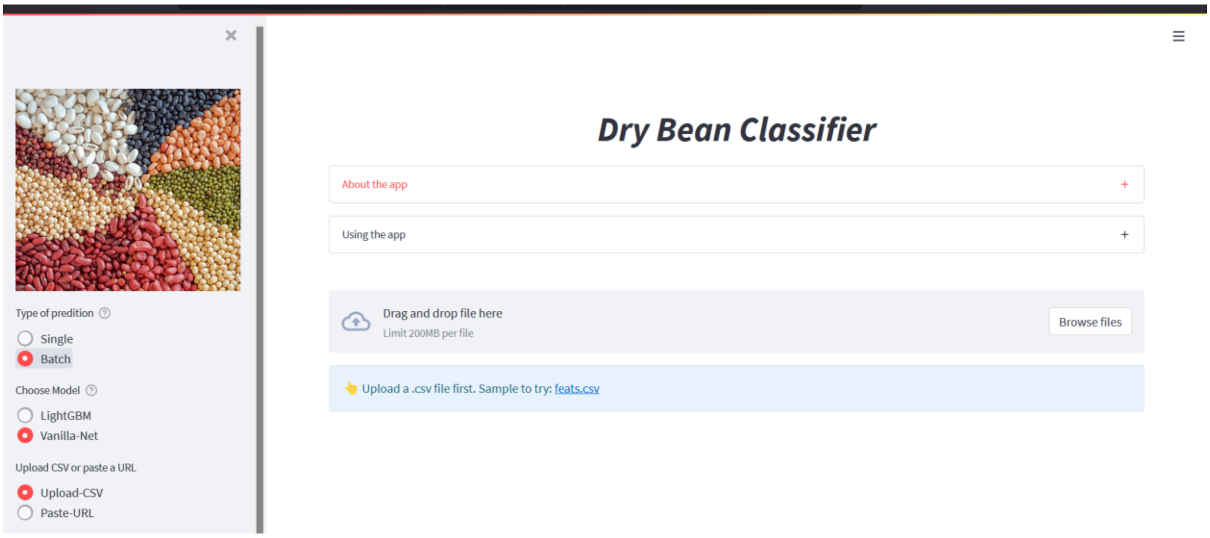
Area	Perimeter	MajorAxisLength	MinorAxisLength
40000	727.877	246.6991625	206.8884621
AspectRatio	Eccentricity	ConvexArea	EquiDiameter
1.192425909	0.544706845	40425	225.6758334
Extent	Solidity	Roundness	Compactness
0.755857899	0.989486704	0.94875453	0.914781514
ShapeFactor1	ShapeFactor2	ShapeFactor3	ShapeFactor4
0.006167479	0.00266414	0.836825218	0.997852072

The form has been pre-filled with default values for ease of demonstration.

2. Batch prediction using .csv file

You can also use a .csv file or a link to a csv file for doing prediction in bulk

Upload CSV directly



Type of predition

○ Single
● Batch

Choose Model

○ LightGBM
● Vanilla-Net

Upload CSV or paste a URL

● Upload-CSV
○ Paste-URL

Dry Bean Classifier

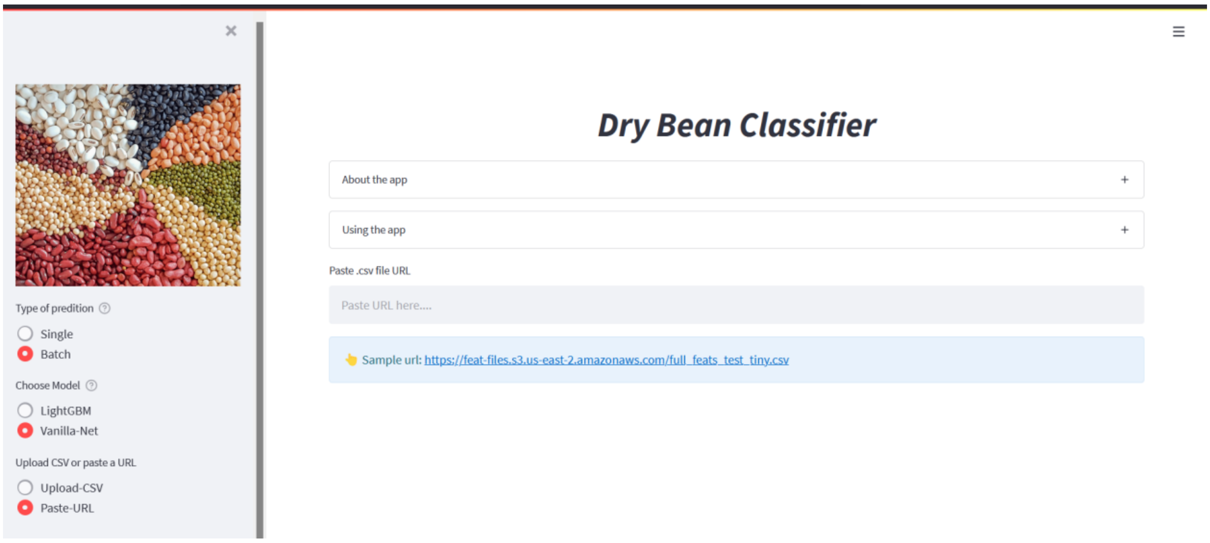
About the app +

Using the app +

Drag and drop file here
Limit 200MB per file

👆 Upload a .csv file first. Sample to try: feats.csv

Paste a link to a CSV file



Type of predition

○ Single
● Batch

Choose Model

○ LightGBM
● Vanilla-Net

Upload CSV or paste a URL

○ Upload-CSV
● Paste-URL

Dry Bean Classifier

Paste .csv file URL

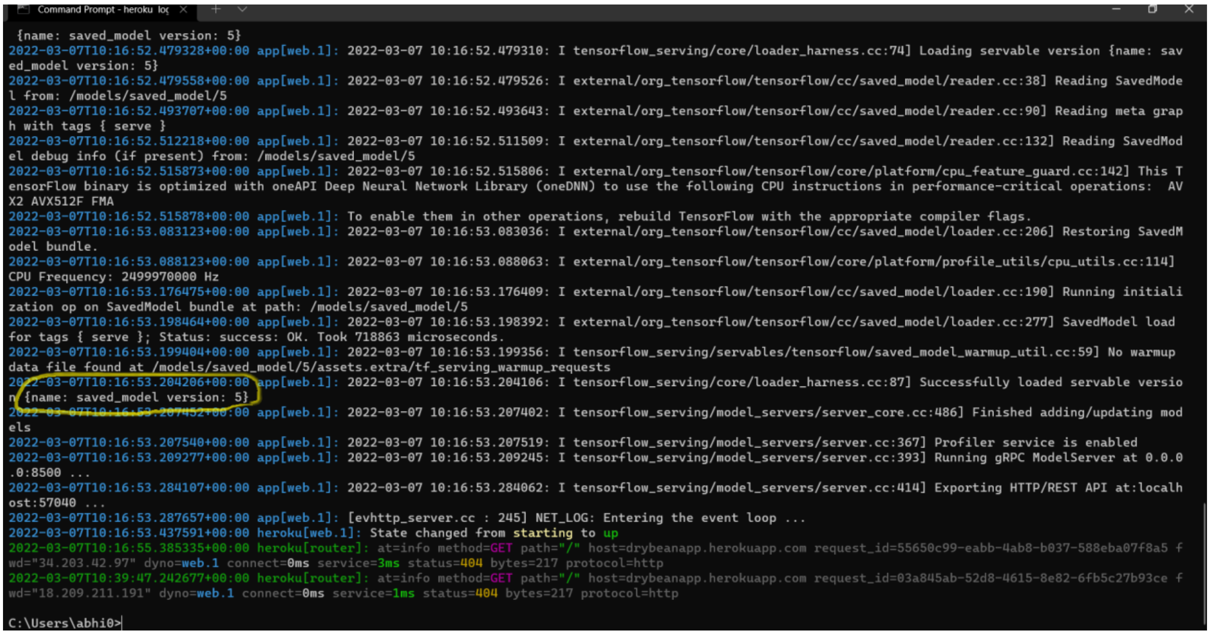
Paste URL here....

👆 Sample url: https://feat-files.s3.us-east-2.amazonaws.com/full_feats_test_tiny.csv

Deploying the Tensorflow model

So, loading the TF model along with the app in Streamlit makes the app slower. So, to solve this I have the Vanilla Net model served at a docker container on Heroku using Tensorflow serving.

The logs of the deployed model look like this, you can see that version 5 of the model is deployed.



```
 {name: saved_model version: 5}
2022-03-07T10:16:52.479328+00:00 app[web.1]: 2022-03-07 10:16:52.479310: I tensorflow_serving/core/loader_harness.cc:74] Loading servable version {name: sav
ed_model version: 5}
2022-03-07T10:16:52.479558+00:00 app[web.1]: 2022-03-07 10:16:52.479526: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:38] Reading SavedMode
l from: /models/saved_model/5
2022-03-07T10:16:52.493707+00:00 app[web.1]: 2022-03-07 10:16:52.493643: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:90] Reading meta grap
h with tags { serve }
2022-03-07T10:16:52.512218+00:00 app[web.1]: 2022-03-07 10:16:52.511509: I external/org_tensorflow/tensorflow/cc/saved_model/reader.cc:132] Reading SavedMod
el debug info (if present) from: /models/saved_model/5
2022-03-07T10:16:52.515873+00:00 app[web.1]: 2022-03-07 10:16:52.515806: I external/org_tensorflow/tensorflow/core/platform/cpu_feature_guard.cc:142] This T
ensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:  AV
X2 AVX512F FMA
2022-03-07T10:16:52.515878+00:00 app[web.1]: To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2022-03-07T10:16:53.083123+00:00 app[web.1]: 2022-03-07 10:16:53.083036: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:206] Restoring SavedM
odel bundle.
2022-03-07T10:16:53.088123+00:00 app[web.1]: 2022-03-07 10:16:53.088063: I external/org_tensorflow/tensorflow/core/platform/profile_utils/cpu_utils.cc:114]
CPU Frequency: 2499970000 Hz
2022-03-07T10:16:53.176475+00:00 app[web.1]: 2022-03-07 10:16:53.176409: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:190] Running initiali
zation op on SavedModel bundle at path: /models/saved_model/5
2022-03-07T10:16:53.198464+00:00 app[web.1]: 2022-03-07 10:16:53.198392: I external/org_tensorflow/tensorflow/cc/saved_model/loader.cc:277] SavedModel load
for tags { serve }; Status: success: OK. Took 718863 microseconds.
2022-03-07T10:16:53.199404+00:00 app[web.1]: 2022-03-07 10:16:53.199356: I tensorflow_serving/servables/tensorflow/saved_model_warmup_util.cc:59] No warmup
data file found at /models/saved_model/5/assets.extra/tf_serving_warmup_requests
2022-03-07T10:16:53.204206+00:00 app[web.1]: 2022-03-07 10:16:53.204106: I tensorflow_serving/core/loader_harness.cc:87] Successfully loaded servable versio
n {name: saved_model version: 5}
2022-03-07T10:16:53.207452+00:00 app[web.1]: 2022-03-07 10:16:53.207402: I tensorflow_serving/model_servers/server_core.cc:486] Finished adding/updating mod
els
2022-03-07T10:16:53.207540+00:00 app[web.1]: 2022-03-07 10:16:53.207519: I tensorflow_serving/model_servers/server.cc:367] Profiler service is enabled
2022-03-07T10:16:53.209277+00:00 app[web.1]: 2022-03-07 10:16:53.209245: I tensorflow_serving/model_servers/server.cc:393] Running gRPC ModelServer at 0.0.0
.0:8500 ...
2022-03-07T10:16:53.284107+00:00 app[web.1]: 2022-03-07 10:16:53.284062: I tensorflow_serving/model_servers/server.cc:414] Exporting HTTP/REST API at:localh
ost:57040 ...
2022-03-07T10:16:53.287657+00:00 app[web.1]: [evhttp_server.cc : 245] NET_LOG: Entering the event loop ...
2022-03-07T10:16:53.437591+00:00 heroku[web.1]: State changed from starting to up
2022-03-07T10:16:55.385335+00:00 heroku[router]: at=info method=GET path="/" host=drybeanapp.herokuapp.com request_id=55650c99-eabb-4ab8-b037-588eba07f8a5 f
wd="34.203.42.97" dyno=web.1 connect=0ms service=3ms status=404 bytes=217 protocol=http
2022-03-07T10:39:47.242677+00:00 heroku[router]: at=info method=GET path="/" host=drybeanapp.herokuapp.com request_id=03a845ab-52d8-4615-8e82-6fb5c27b93ce f
wd="18.209.211.191" dyno=web.1 connect=0ms service=1ms status=404 bytes=217 protocol=http

C:\Users\abhi0>
```

So, when you click on predict button the app is just making calls to this endpoint! The interesting thing about it is that all you need to do is just hit this endpoint for the prediction, so you can do it on your own without the app by a simple python script.