# ML Deployment – Sports News Headlines Classifier

* Build Package:
  + Project folder path: C:\Users\Aiden\Documents\Data\_Science\_Stuff\sf\_Data\_Science\_Stuff\Projects\06\_ML\_Deployment\01\_Build\_Package
  + I have taken a copy of the code from the actual project’s folder path:
  + C:\Users\Aiden\Documents\Data\_Science\_Stuff\sf\_Data\_Science\_Stuff\Projects\01\_SportsNewsClassifier\final\_code
  + I have changed some of the code:
    - Code to process data and build model is saved in logistic\_regression\_model folder.
    - instead of scraping data, we load CSVs – **this is to ensure the best practice of reproducibility in our ML pipeline**
    - I have created requirements.txt and test\_requirements.txt for installing dependencies via Tox (we are not using conda!!)
    - I have created validation.py and predict.py that makes a prediction on a batch of data
    - I have used pytest instead of unittest. Tests are saved in the folder tests. The conftest.py loads the testing dataset which is used on the pytests.
    - After deploying to Heroku it seemed that using OS didn’t work for project parameters, so I changed the code base back to using pathlib.
  + Training data = Sports\_News\_09\_09\_2021.csv
  + Testing data = Sports\_News\_03\_06\_2021.csv
  + As mentioned above, I am using tox to run the pipeline:
  + tox.ini file contains commands to run the pipeline
  + most of the code was copied from Udemy course section 5 tox.ini
  + I also copied over the mypy.ini file as these remove warnings when running commands in tox.ini.
  + pytest was producing deprecation warnings so I added another file called pytest.ini to remove these warnings when running tox
  + At this point the package is ready to be built
  + I copied over the pyproject.toml file from Udemy and at the bottom I changed the line\_length to 135
  + I also copied over setup.py and made changes:
    - Package meta-data
    - Instead of pathlib, I have used os to locate directories
  + I created a file called VERSION that specifies the version of the model.
  + Lastly, I built the python package for the model (tid-logistic-regression-model)
    - I added init.py file to the folder logistic\_regression\_model. Without this file, the package will not work in the next section
* **FastAPI**
  + FastAPI loads after running tox -e run
    - Ensure correct version of model is in requirements.txt
  + localhost:8001 to view app
  + /docs to execute test prediction
  + Can also use requests module in Python to make a prediction:
* **Deploy to Heroku**
  + Create Heroku app from cli: floating-anchorage-18633
  + At this point we will need a git repository (create development branch in git and push all code there!)
  + git subtree push --prefix 02\_Deploy\_REST\_API main
  + this should deploy the app – go back to Heroku and open app. Can check out /docs and execute prediction.
  + Can test API using requests in Python – similar url to localhost above, replace localhost with Heroku url
* **CI/CD** 
  + Using CircleCI we need to automate the following process:
    - Train and test new model, then publish to Gemfury
    - Deploy app to Heroku
  + I have taken a copy of the model package and saved into 03\_CircleCI, in the subfolder model-package:
    - We have the code to train, test and save the model (logistic\_regression\_model folder).
    - requirements, test, MANIFEST, mypy.ini, setup.py files etc.
    - We now have a new file called publish\_model.sh
      * This will be used in the tox file to upload our model package to Gemfury.
      * I have copied this file from the Udemy course without any changes
    - We add new jobs to the tox.ini file
    - [testenv:publish\_model] – this runs the main.py file and then publishes models to Gemfury
    - We also add in the passenv GEMFURY\_PUSH\_URL (line 17)
  + We copy over our FastAPI code and save to the folder sports-classifier-api
    - We add the passenv to the tox.ini file PIP\_EXTRA\_INDEX\_URL so that we can get our published model from Gemfury.
    - We also add an extra line at the top for the requirements.txt file so that the model package is fetched from Gemfury.
  + We can now create our config.yml file that will automate the process of our entire pipeline.
    - Copy config.yml from Udemy course – I only kept the jobs that train, test and publishes model, as well as push to Heroku
    - For the app to be deployed to Heroku, the Udemy course required branch for master only (line 99-105). I have added dev\_aiden and development so any changes on these branches will run the CI/CD process.
  + I added the env variables for Heroku and CircleCI (note that we need to set project up in CircleCI at this point)
  + Commit these changes to dev\_aiden and push to origin which should automatically start the CI/CD pipeline. Note that if style checks etc. fail, then run locally on terminal and then re-commit and push to origin. Only then will the tests pass.
  + Once the tests have passes then deployment to Heroku will occur. Note that at this point the model package is not on Gemfury! So deployment may work correctly, but the process is not fully automated.
  + To publish model to Gemfury, increment the version and then update requirements to new version. Then using git tag, commit these changes and push to origin. CircleCI will then publish model to Gemfury with new version.
  + **It is always best to first run tox locally before pushing changes to git**