1. Do the model iterations and finalize all the steps for data preprocessing and model building.
2. Divide your script into three functions:
   1. Data preprocessing and train/test split
   2. Model training with hyper-parameters
   3. Get Model metrics

Project\_folder\_structure:

1. First set up azure environment:

‘Environment\_setup’ folder has 2 files:

1. cloud-environment.json: No input given. It has all the items to be created in azure ml workspace along with the name
2. iac-create-environment-pipeline-arm.yml: No changes made. It takes following inputs to define names in azure ml workspace. This should be given as pipeline variables with same group name as ‘mlops-wsh-vg’:
   1. AZURE\_RM\_SVC\_CONNECTION
   2. LOCATION
   3. RESOURCE\_GROUP
   4. BASE\_NAME
   5. WORKSPACE\_NAME
   6. AZURE\_ML\_WORKSPACE\_CONNECTION
3. package\_requirement: Modify the requirements.txt accordingly.
4. Data: keep the data for training the model
5. [project\_param.yaml](http://localhost:8888/edit/Documents/my_deplyment_learning/titanic_model_MLOPS/project_param.yaml): change parameters here according to project. These will be used in train\_aml.py file
6. Training the model:
   1. conda\_dependencies.yml: The dependencies defined in this file will be automatically provisioned for managed runs. These include runs against the localdocker, remotedocker, and cluster compute targets. Just check once if all required packages are mentioned in it.
   2. Train.py: define Data preprocessing and train/test split, Model training with hyper-parameters and Get Model metrics functions here according to your business problem.
   3. train\_test.py: modify this file according to the function defined in tain.py file to run the unit testing of these functions.
   4. parameters.json: hyper tuning parameters of the model
   5. train\_config.runconfig : it is run configuration file. It has all details about env, history/run logs and data reference. Replace 'DATA\_FOLDER\_NAME' with the name of folder created in datastore in Azure while uploading the data. E.g.: titatnic
   6. train\_aml.py: It registers the dataset from the uploaded one in azureblobstorage and create version. Run the model and save everything in output folder along with logging things at run. No need to change anything here.
7. Deployment:
   1. [aciDeploymentConfigStaging.yml](http://localhost:8888/edit/Documents/my_deplyment_learning/titanic_model_MLOPS/deployment/aciDeploymentConfigStaging.yml): req for staging VM machine. No change
   2. [aksDeploymentConfigProd.yml](http://localhost:8888/edit/Documents/my_deplyment_learning/titanic_model_MLOPS/deployment/aksDeploymentConfigProd.yml): req. for prod incase using kubernet. No change needed.
   3. inferenceConfig.yml: used for deployment. Havepath to scoreing file and its configuration
   4. scoringConfig.yml: similar to conda\_dependecy file. Defining the env
   5. score.py: it acts like fastAPI file which will be used for model testing/scoring. No changes needed.
8. Test staging/ production:
   1. conftest.py: have scoreurl and scorekey to pass on to stag\_test and prod\_test.py files. No changes needed.
   2. Stage\_test.py: this will test if score.py function in stage env. Same thing for prod\_test in prod env. Given the relevant dataset input.