ML Developer Guide

Perform EDA and initial model experimentation

Prior to deployment we need to be able to demonstrate that a machine learning model is a viable solution to a business problem.

EDA and model experimentation should be carried out using notebooks on databricks with all findings written down within the the notebook.

Ensure you have also completed the Business Understanding, Data Understanding and Modelling pages for the project on Confluence.

When you're satisfied with initial ML experimentation (e.g. validated that a model with reasonable performance can be trained on your dataset and agreed model performance levels with stakeholders) and are ready to deploy production training/inference pipelines, move to the initial setup phase documented below.

Initial Project Setup

ML Repos come with a template that provides the preferred structure for a new ML project. Using the template generates the required files and code for training, validating and deploying a simple regression model using simulated data.

This code can then be adapted to your ML problem, reducing the time to deploy the project.

To get started and generate a new project directory, follow the steps below:

1. Clone the repo to your local machine. (**Only do this if you've never cloned the repo before!**)
   - (Optional) Contact `@Samuel.Thomas13` or `@Andrew Witherley` if a new business area repo needs to be set up
   - Open Windows command prompt and navigate to the location you want to save the repo
   - Enter `git clone <repo url>`
   - You should now see the repo folders in your local drive
2. Using windows command prompt or the terminal in VS Code, navigate to the repo root directory
3. Enter `git checkout main` to ensure you are on the main branch
4. Enter `git pull` in the command prompt to ensure you have the latest code updates
5. Enter `git checkout -b <name_of_new_branch>` to create a new branch to hold the project. Convention is to name this `feature/<add_new_project_name_to_repo>`
6. Enter the command `databricks bundle .\new_project_template\` and follow the prompts to autopopulate fields in the resulting files
7. You should now be able to see a new project directory in the repo root folder
8. Subsequent sections explain how to adapt the example code to your ML problem and quickly get started iterating on model training code.

Iterating on ML code

Now that the project structure is populated you may transplant code from the exploration notebooks into the relevant sections of the template files.

The following is a non-exhaustive list of best practices:

- When moving code across you only need to consider the code used for the modelling elements of the project. Usually this will be the data acquisition, feature engineering, training and inference code.
- You should only include visualisation code where it is a required output for the stakeholders or for regulatory reasons.
- Modularise your code into small, testable functions.
- Write unit tests for your new functions. Use the tests folder as a guide for how to structure your tests
- The `.py` files in the notebooks directories act as pipelines tying the individual functions together to create a feature engineering pipeline or a training pipeline. You will need to import the functions and combine them together within these notebooks.
- For writing code you can either use Databricks repos or develop using your local machine. This is personal preference but it is generally easier to work with mutliple files using an IDE or editor such as VS Code.
- When reading production data it is best practice to use Federated Queries in Unity Catalog. This means there is no need to use service accounts.
- When writing results back to Snowflake, you will need to use the Data Science service accounts.
  - **YOU MUST WRITE ONLY TO THE DATA SCIENCE DATABASES (PPD_DS, PRD_DS) UNLESS AGREED OTHERWISE WITH THE BUSINESS AND HEAD OF DATA SCIENCE!**
  - Use the code below to get the DS credentials:
  - ```python
    def get_and_set_data_science_sf_options(env: str, schema: str) -> tuple[dict, dict]:
        """
        Reads the service account credentials and sets up the options for connecting to Snowflake with the datascience service account

        Args:
            env: which environement we are running and therefore which key values pairs to get

        Returns:
            ds_options: a dictionary containing the options required to access datascience bds in snowflake
        """
        if env in ('ppd', 'pps'):
            env = 'ppd'
            database = 'PPD_DS'
            warehouse = 'PPD_DS_LOAD_WH'
        elif env == 'prd':
            env = 'prd'
            database = 'PRD_DS'
            warehouse = 'PRD_DS_LOAD_WH'

        print(database)
        # Get the SF service account details
        ds_user = dbutils.secrets.get(scope='kv_snowflake', key='datascience-{}-snowflake-sa-user'.format(env))
        ds_pwd = dbutils.secrets.get(scope='kv_snowflake', key='datascience-{}-snowflake-sa-user-pwd'.format(env))

        ds_options = {
                "host": "threemobile.west-europe.privatelink.snowflakecomputing.com",
                "user": ds_user,
                "password": ds_pwd,
                "sfWarehouse": warehouse,
                "database": database,
                "schema": schema
            }

        return ds_options
    ```
    ```

- Once the peer review and head of DS review are complete, a Jira ticket can be sent to Data Engineering to set up a view in the business' BDS that references the output table for your model

Running your code

In order to check that your code works you will need to run it. Our recommended practice is to run code on Databricks. This reduces the dependencies on the different software and hardware configurations of our laptops, as well as ensuring we harness the compute and storage provided by databricks.

There are two ways to move code from your local machine to databricks: Databricks Repos and Databricks Bundles.

Databricks Repos

Using Databricks Repos is a useful way to move code to Databricks in order to test and validate isolated functions and single pipelines (e.g. the training pipeline in Train.py)

Simply follow the UI workflow to connect the dev workspace to the Azure repository

From there you can spin up a cluster and run the different modules as required.

It is recommended to run unit tests using Databricks Repos

Databricks Bundles

Bundles are a way of packaging and deploying ML code together with cluster resource configs to the dev workspace.

Bundles are best used when you want to deploy and test an end-to-end pipeline (e.g. feature engineering -> model training -> inference).

Once you have written the code and are ready to test the pipelines, the resource and code can be deployed as follows:

1. Open Windows command prompt or a terminal in your IDE.
2. Navigate the directory containing your project's files.
3. Enter the command `databricks bundle validate` to validate the contents of the .yml files in the `project_name\resources` directory. You should see a print out of json schemas and a message `Validation OK!` f the bundle has validated correctly. If there are any errors then update the .yml files appropriately.
4. If no errors are present from the previous step then enter `databricks bundle deploy -t ppd` to deploy the code, job config and cluster resource config to the PPD workspace
5. Navigate to the Workflows section in the Databricks UI and you should see the jobs defined by the resources files. Run and test these workflows using the UI.
6. You can also run the workflow from the CLI using command `databricks bundle run <job_name> -t ppd`
7. Tidy up the workspace by entering `databricks bundle destroy -t ppd` on your local machine (**ensure you are in the project directory before running this!**)
8. For further details on how to set up and configure databricks project resources, see ML resource config guide.

For further details on how to set up and configure databricks project resources, see ML resource config guide.

Next Steps

Once your project code and resources are ready to be deployed to production (or if you are updating an existing project), follow Submitting a Pull Request to submit your code for testing and production deployment.