Feature Engineering

Connect to the Workspace

In [2]:
```
from azure.ai.ml import MLClient
from azure.identity import DefaultAzureCredential
from azureml.core import Workspace, Dataset
```

In [3]:
```
# Get credentials
credential = DefaultAzureCredential()

# Load the workspace
ws = Workspace.from_config()

SUBSCRIPTION_ID = ws.subscription_id
RESSOURCE_GROUP = ws.resource_group
WORKSPACE_NAME = ws.name

# Connect to the workspace
ml_client = MLClient(
    credential=credential,
    subscription_id=SUBSCRIPTION_ID,
    resource_group_name=RESSOURCE_GROUP,
    workspace_name=WORKSPACE_NAME
)
```

Configure Environment

In [2]:
```
from azure.ai.ml.entities import Environment

env_name = "dev-env"
env_version = '1.1'
env = None
conda_file_path = 'conda-dev.yaml'

try:
    env = ml_client.environments.get(env_name, version=env_version)
    print(f"Found existing environment: {env.name}")
except Exception as e:
    print(f"Environment {env_name} not found. Creating a new environment.")


    env = Environment(
        name=env_name,
        description="Custom environment for Modelling pipeline",
        tags={"scikit-learn": "1.3.0"},
        version=env_version,
        conda_file=conda_file_path,
        image="mcr.microsoft.com/azureml/openmpi3.1.2-ubuntu18.04:latest",
    )

    env = ml_client.environments.create_or_update(env)

print(
    f"Environment with name {env.name} is registered to workspace, the environment version is {env.version}"
)
```

Create a feature-engineering script

In [6]:
```
import os

fe_src_dir = "./src/feature_engineering"
os.makedirs(fe_src_dir, exist_ok=True)
```

In [7]:
```
%%writefile {fe_src_dir}/create_features.py

"""
Add new features to the processed data.

Note: Executed after `preproccessing/preprocess.py`.
"""



import pandas as pd
import mlflow
import argparse



def parse_args():
    parser = argparse.ArgumentParser()
    parser.add_argument("--input_data", type=str, help="path to input data")
    parser.add_argument("--output_data", type=str, help="path to output data")

    return parser.parse_args()


def main(args):
    # Load the preprocessed data
    df = pd.read_parquet(args.input_data)

    # Create a new feature: "Has"
    df["Has"] = (
            df["Date"] > df["Date in"]
    )

    # Remove the dates since we don't need them anymore
    dates = df.select_dtypes(
                include=["datetime64", "datetime", "timedelta64", "timedelta"]
            ).columns
    df.drop(columns=dates, inplace=True)

    # Save the engineered data
    df.to_parquet(args.output_data, index=False)

    # Log the results
    mlflow.log_metric("new_features", df.columns.tolist())
    mlflow.log_metrics("n_samples", df.shape[0])
    mlflow.log_metrics("n_features", df.shape[1])

    print("New features added successfully!")


if __name__ == "__main__":
    print("Running script to create new features...")

    mlflow.start_run()

    args = parse_args()
    main(args)

    mlflow.end_run()
```

```
Overwriting ./src/feature_engineering/create_features.py
```

Run the script

In [8]:
```
from azure.ai.ml import command, Input, Output, UserIdentityConfiguration, ManagedIdentityConfiguration
from azure.ai.ml.entities import Data
from azure.ai.ml.constants import AssetTypes, InputOutputModes
```

Prepare the input and output data

In [9]:
```
# Data path

datastore_root_folder = 'data'
datastore_url = f'azureml://subscriptions/{SUBSCRIPTION_ID}/resourcegroups/{RESSOURCE_GROUP}/workspaces/{WORKSPACE_NAME}/datastores/workspaceblobstore/paths/{datastore_root_folder}'

input_file_path = 'processed/processed.parquet'
input_data_path = f'{datastore_url}/{input_file_path}'

output_file_path = 'engineered/engineered.parquet'
output_data_path = f'{datastore_url}/{output_file_path}'


inputs = {
    'input_data': Input(type=AssetTypes.URI_FILE, path=input_data_path, mode=InputOutputModes.RO_MOUNT),
}

# To create a data asset from the output sepcify a name
# If version is not specified, it is automatically incremented
outputs = {
    "output_data": Output(
        type=AssetTypes.URI_FILE,
        path=output_data_path,
        mode=InputOutputModes.RW_MOUNT,
        name = "dataset_engineered",
        # version = "1",
)
}
```

Configure the command

In [1]:
```
from random import choices
from string import ascii_uppercase, digits

compute_name = "VAZ-DS2"

random_string = ''.join(choices(ascii_uppercase + digits, k=7)).lower()
display_name = "feature-engineering-" + random_string

job = command(
    inputs=inputs,
    outputs=outputs,
    compute=compute_name,
    code="./src/feature_engineering/",  # location of source code
    command="python create_features.py --input_data ${{inputs.input_data}} --output_data ${{outputs.output_data}}",
    environment=f"{env.name}@latest",
    experiment_name="modelling-from-notebook",
    display_name=display_name
```

```
)

# Run the job
ml_client.create_or_update(job)
```