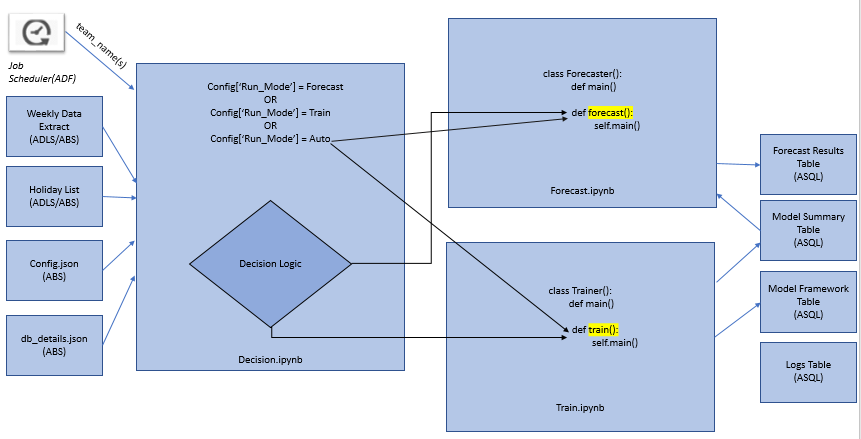
**Code Flow- High Level**



ADF – Azure Data Factory,

ADLS – Azure Data Lake Services,

ABS – Azure Blob Storage,

ASQL – Azure SQL

**Decision.ipynb**

**This is the master notebook that will be called by the databricks job.**

Takes ‘team\_name’ as input argument. Below is the main function of this notebook that starts the processing

if \_\_name\_\_ == ‘main’:

-Initialization block

-Initialize Logger Object

-Record Code Run Start Time

-Accept Team Name argument from job scheduler

-Mount Drives (ABS/ADLS)

-Run\_Mode Conditional Check Block

if run\_mode = train:

Initialize Trainer Object and call Trainer.train(team, code\_run\_start\_utc)

elif run\_mode = forecast:

Initialize Forecaster Object and call Forecaster.forecast(team, code\_run\_start\_utc)

elif run\_mode = auto:

mode = decision\_box(team, code\_run\_start\_utc)

if mode = forecast:

Initialize Forecaster Object and call Forecaster.forecast(team, code\_run\_start\_utc)

elif mode=full refresh:

Initialize Trainer Object and call Trainer.train(team, code\_run\_start\_utc)

Initialize Forecaster Object and call Forecaster.forecast(team, code\_run\_start\_utc)

**Train.ipynb**

This notebook is used for a full-training of all the models for a given team

This script follows an Object-oriented design where there is one class, called Trainer which has all the required functions as its methods

Trainer.train() -> calls the main() method

Trainer.main()

:Takes team\_name and code\_run\_start\_utc as input arguments

:Returns: None

Function calls:

- Instantiates object of the Logger class by passing team\_name and prefix=’train’ for defining a unique log file path to save the logs to

- get\_env() to fetch environment name from config file/tags (rnd/test/prd)

- get\_table\_names(env) to fetch table names for the given environment from the db\_details.json config file

- train\_ADE and train\_ADE\_standalone flags fetched from config file which indicate whether and in which mode to train the ADE (standalone training/on-line training)

- conditional blocks to train either (i) only the base models [if train\_ADE = False(0)] (ii) only the Ensemble model [if train\_ADE = True(1) and train\_ADE\_standalone=True(1)] or (iii) base models followed by Ensemble model if [if train\_ADE = True(1) and train\_ADE\_standalone=False(0)]

-conditional check to see if df\_summary (dataframe holding model training summary results from all models) has non-zero number of rows. If not, then terminate training script as no model to choose from. If yes, then continue with the rest of the code

-df\_best = model\_selector(df\_summary) This takes a dataframe of say 37 models (1 row per model) and returns a dataframe of the same number of rows (37) that is sorted in the order of the best model selection logic, with an additional ‘indicator’ column called BEST\_MODEL which holds the model rank from 1 to 37

-Round off the predictions in the framework dataframe

-self.write\_results\_to\_file() Saves the summary and framework audit dataframes as flat csv files to ABS/ADLS

-self.insert\_results\_to\_db() saves the summary and framework audit dataframes to the ASQL database tables as per the current environment (rnd/test/prd)

-logger.log\_file\_to\_db() Picks up the saved train log file from ABS/ADLS, reads its content into pandas and subsequently spark dataframe and using spark sql, writes the logs to an ASQL table for audit trail review.

**Forecast.ipynb**

This notebook is used for performing forecasts with a given set of models for a given team

This script follows an Object-oriented design where there is one class, called Forecaster which has all the required functions as its methods

Forecaster.forecast() -> calls the main() method

Forecaster.main()

:Takes team\_name and code\_run\_start\_utc as input arguments

:Returns: None

Function calls:

- Instantiates object of the Logger class by passing team\_name and prefix=’forecast’ for defining a unique log file path to save the logs to

- get\_env() to fetch environment name from config file/tags (rnd/test/prd)

- get\_table\_names(env) to fetch table names for the given environment from the db\_details.json config file

-prediction\_preiod = config['prediction\_period'] Gets the prediction period (in weeks) to forecast into the future

-top\_n\_models = config.get('num\_forecasters',1) Gets the number of models to use to make the forecast(s). Defaults to 1 if not provided in config file

- model\_train\_results = self.get\_model\_train\_results(team\_name, tables) Gets the training summary results saved in the summary\_results table from the last training process tht was performed on the given team\_name. If the result set of the query on the database returns an empty set or if there is some error in querying the table, this functions logs an exception and causes this script to exit, as there is no info available on any model to make a forecast

Otherwise, it returns a dictionary of the model training summary results from the last refresh

- generate\_forecast\_results(model\_train\_results, top\_n\_models=1)

:returns dataframe of appended forecasts from several models

Taking the output from the previous method (model\_train\_results ) as an input, this function runs a while loop on that input dataframe for min(top\_n\_models, number of rows in model\_train\_results) calling the ‘model’ methods of each of the model classes and passing the input df + prediction\_period number of rows in order to make the forecasts. At the end of each iteration, the forecasts are appended together

-self.write\_results\_to\_file() Saves the forecast results dataframe as flat csv file to ABS/ADLS

-self.insert\_results\_to\_db() saves the forecast result dataframe to the ASQL database table forecast\_results as per the current environment (rnd/test/prd)

- logger.log\_file\_to\_db() Picks up the saved forecast log file from ABS/ADLS, reads its content into pandas and subsequently spark dataframe and using spark sql, writes the logs to an ASQL table for audit trail review.

**Code Folder Structure:**

└───**modular**

│

└───**installations**

| │─── pmdarima==1.6

| │─── statsmodels==0.10.1

| |─── scipy==1.3.1

│

└───**src**

│─── decision

| ─── forecast

│─── train

│

├───**data**

│ │─── preprocessing

| | |\_get\_preprocessed\_df()

| | |\_get\_preprocessed\_df\_forecast()

│ │─── reporting

| |\_\_Logger()

|

│ │─── sql\_insert

| │── sql\_read

│

├───**models**

│ │───Model\_parallel

| |───Model\_ADE

│ │───Model\_arima

│ │───Model\_baseline

│ │───Model\_des\_wo\_damping

│ │───Model\_des\_w\_damping

│ │───Model\_sarima

│ │───Model\_ses

│ │───Model\_sma

│ │───Model\_tes\_wo\_damping

│ │── Model\_tes\_w\_damping

│ │──Model\_weekly\_trend

│ │──Model\_wma12

│ │──Model\_wma60

| │───model\_selection

│ │───model\_object\_picker

│

├───**utils**

│ │──training\_helper

│──db\_utils

|\_\_get\_env()

|\_\_get\_table\_names(env)

|\_\_get\_jdbc\_connection(env)

**Configuration Files:**

**config.json**

This is the main (driver) configuration file that initializes global parameters to start the code runs

{

    "env":"rnd", Azure Environment Indicator. (rnd/nonprd/prd)

    "data\_file": "/mnt/csi\_forecasting/data/curated/workstack\_forecasting/{datetime}\_csi\_fo recasting\_demand\_file.parquet", ADLS Path to data extract parquet file. {datetime} is a placeholder that gets replaced inside the code with the code run start date

    "holiday\_file": "/dbfs/mnt/csi\_forecasting/data/Holiday\_List.csv",

    ADLS Path to file containing list of bank holidays

    "run\_mode":"train", Mode to run the code in – ‘train’ (full-refresh) or ‘forecast’

    "num\_forecasters":2, If run\_mode = ‘forecast’, then this parameter indicates how many models to forecast using

    "num\_iterations":53, Number of iterations (weeks) of walk-forward to use while training

    "evaluation\_period":8, Number of weeks to forecast ahead

    "prediction\_period":8, Number of weeks to forecast ahead while training

    "train\_ADE":1, Flag indicating whether (1) or not(0) to train the Ensemble Model

    "train\_ADE\_standalone":1, If train\_ADE = 1, this flag indicates whether to train the Ensemble in standalone (1) or on-line (0) mode

    "holiday\_treatment\_logics": ["None", "Nearest Neighbour", "Weekday Distribution"],

List of bank holiday treatment logics to apply while training

"models\_to\_train": [

                        "Baseline",  "Arima", "SMA",

                        "DES\_w\_damping", "DES\_wo\_damping",

                        "TES\_w\_damping", "TES\_wo\_damping",

                        "WMA12", "WMA60",

                        "Sarima", "SES",

                        "Weekly\_Trend"

                        ],

Full List of Base Models to train

"num\_models\_train":12, Number of Base Models from the above list to train

    "models\_to\_drop\_for\_ADE": ["WMA12", "WMA60", "Weekly\_Trend", "ADE"]

    Full List of base models to discount while training the ensemble

}

**db\_details.json**

This is a global, environment-agnostic configuration file that contains db connection details and table names of databases across all Azure environments to be used in the project. The **env** parameter from driver config file is used to query this config file for the relevant details

[

{

    "env":"rnd", Environment Name, used as a query key (rnd/nonprd/prd)

    "connection\_properties": {

        "jdbcHostname": "asqlserver-mg-rnd1-t1-weu-csiopfore-01.database.windows.net",

        "jdbcDatabase": "asqldb-mg-rnd1-t1-weu-csiopfore-01",

        "jdbcPort": 1433,

        "username": "\*\*\*\*",

        "password": "\*\*\*\*"

        }, ASQL Database Connection Details for the relevant environment (username and password parameters will be enncrypred)

    "tables": {

        "train\_summary":"dbo.PG\_TEST\_CSI\_FORECASTING\_TRAIN\_SUMMARY",

        "train\_framework":"dbo.PG\_TEST\_CSI\_FORECASTING\_TRAIN\_FRAMEWORK",

        "forecast\_results":"dbo.PG\_TEST\_CSI\_FORECASTING\_FORECAST\_RESULTS",

        "logs":"dbo.PG\_TEST\_CSI\_FORECASTING\_CODE\_RUN\_LOGS"

          } Table Names for tables in the relevant environment

},

{

    "env":"nonprd",

    "connection\_properties": {

        "jdbcHostname": "",

        "jdbcDatabase": "",

        "jdbcPort": 1433,

        "username": "",

        "password": ""

        },

    "tables": {

        "train\_summary":"dbo.SR\_CSI\_FORECASTING\_TRAIN\_SUMMARY",

        "train\_framework":"dbo.SR\_CSI\_FORECASTING\_TRAIN\_FRAMEWORK",

        "forecast\_results":"dbo.SR\_CSI\_FORECASTING\_FORECAST\_RESULTS",

        "logs":"dbo.SR\_CSI\_FORECASTING\_CODE\_RUN\_LOGS"

          }

},

.

.

.

{ similar details for other environments}

.

.

.

]