qml4omics

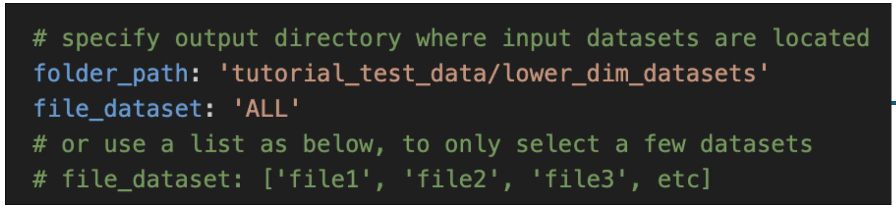
Code overview

So, what does the config file do?

- **It controls everything!**
  - Parameters passed as arguments throughout the code base

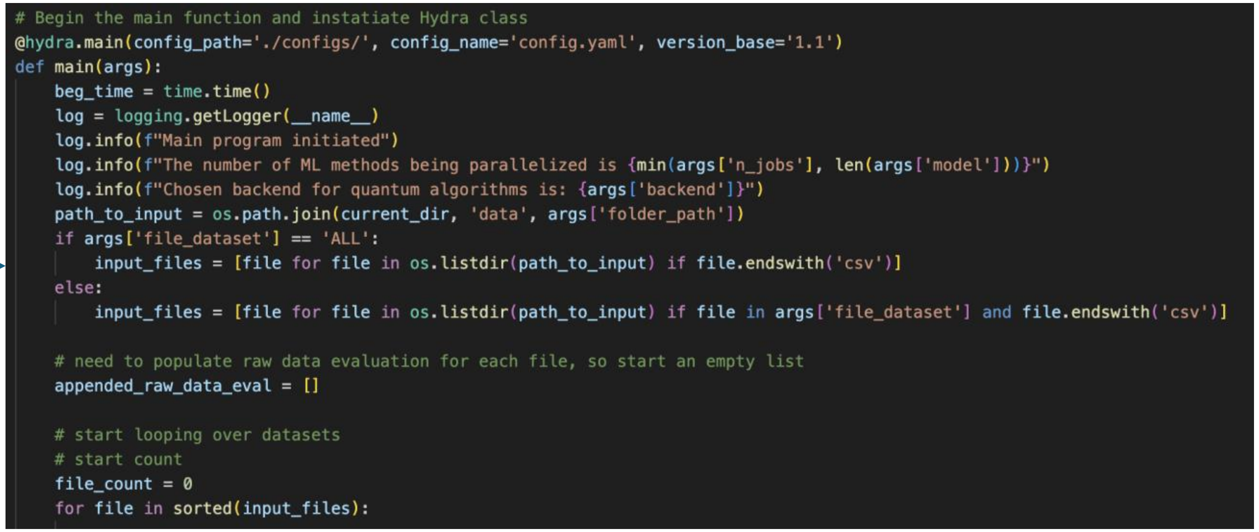
1) Input data sets

config.yaml



```yaml
# specify output directory where input datasets are located
folder_path: 'tutorial_test_data/lower_dim_datasets'
file_dataset: 'ALL'
# or use a list as below, to only select a few datasets
# file_dataset: ['file1', 'file2', 'file3', etc]
```

qml4omics-profiler.py (main function)

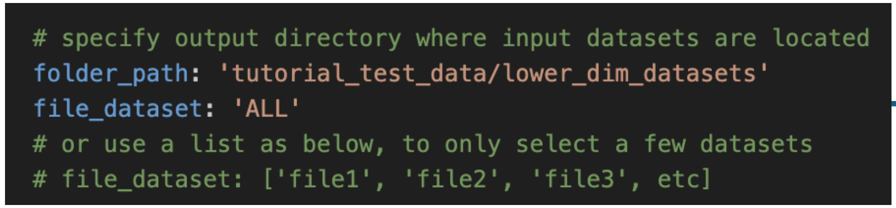


```python
# Begin the main function and instatiate Hydra class
@hydra.main(config_path='./configs/', config_name='config.yaml', version_base='1.1')
def main(args):
    beg_time = time.time()
    log = logging.getLogger(__name__)
    log.info(f"Main program initiated")
    log.info(f"The number of ML methods being parallelized is {min(args['n_jobs'], len(args['model']))}")
    log.info(f"Chosen backend for quantum algorithms is: {args['backend']}")
    path_to_input = os.path.join(current_dir, 'data', args['folder_path'])
    if args['file_dataset'] == 'ALL':
        input_files = [file for file in os.listdir(path_to_input) if file.endswith('csv')]
    else:
        input_files = [file for file in os.listdir(path_to_input) if file in args['file_dataset'] and file.endswith('csv')]

    # need to populate raw data evaluation for each file, so start an empty list
    appended_raw_data_eval = []

    # start looping over datasets
    # start count
    file_count = 0
    for file in sorted(input_files):
```

So, what does the config file do?

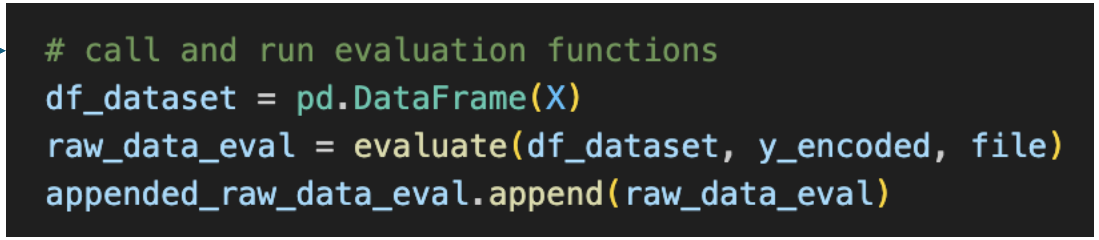
2) Complexity evaluation (on raw and embedded data)

config.yaml

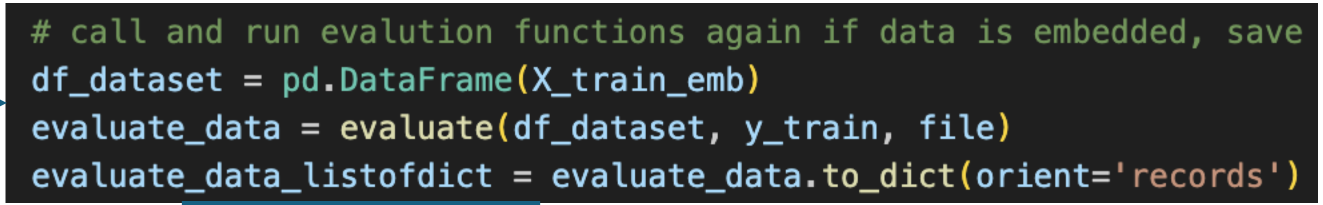


```yaml
# specify output directory where input datasets are located
folder_path: 'tutorial_test_data/lower_dim_datasets'
file_dataset: 'ALL'
# or use a list as below, to only select a few datasets
# file_dataset: ['file1', 'file2', 'file3', etc]
```

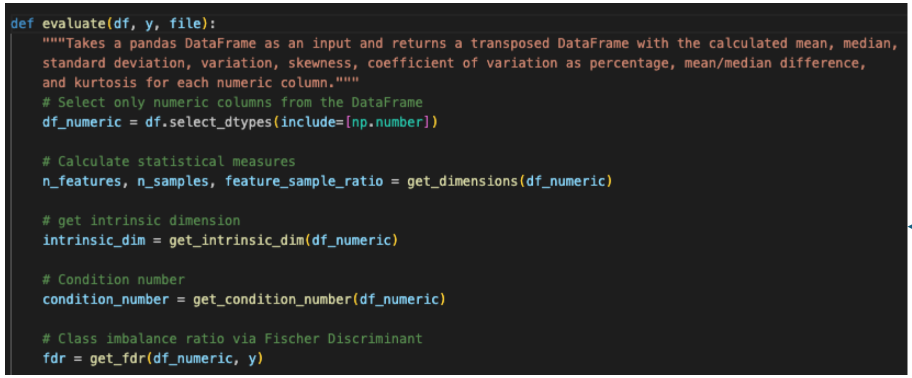
qml4omics-profiler.py (main function)



```python
# call and run evaluation functions
df_dataset = pd.DataFrame(X)
raw_data_eval = evaluate(df_dataset, y_encoded, file)
appended_raw_data_eval.append(raw_data_eval)
```



```python
# call and run evalution functions again if data is embedded, save
df_dataset = pd.DataFrame(X_train_emb)
evaluate_data = evaluate(df_dataset, y_train, file)
evaluate_data_listofdict = evaluate_data.to_dict(orient='records')
```



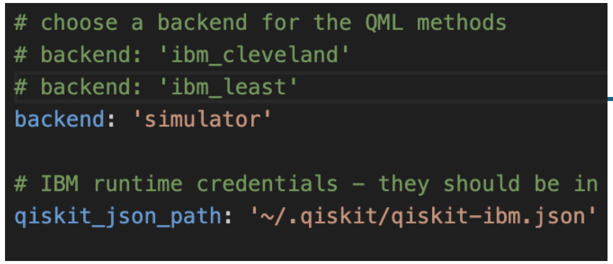
```python
def evaluate(df, y, file):
    """Takes a pandas DataFrame as an input and returns a transposed DataFrame with the calculated mean, median,
    standard deviation, variation, skewness, coefficient of variation as percentage, mean/median difference,
    and kurtosis for each numeric column."""
    # Select only numeric columns from the DataFrame
    df_numeric = df.select_dtypes(include=[np.number])

    # Calculate statistical measures
    n_features, n_samples, feature_sample_ratio = get_dimensions(df_numeric)

    # get intrinsic dimension
    intrinsic_dim = get_intrinsic_dim(df_numeric)

    # Condition number
    condition_number = get_condition_number(df_numeric)

    # Class imbalance ratio via Fischer Discriminant
    fdr = get_fdr(df_numeric, y)
```

qml4omics/evaluation/dataset_evaluation.py

So, what does the config file do?

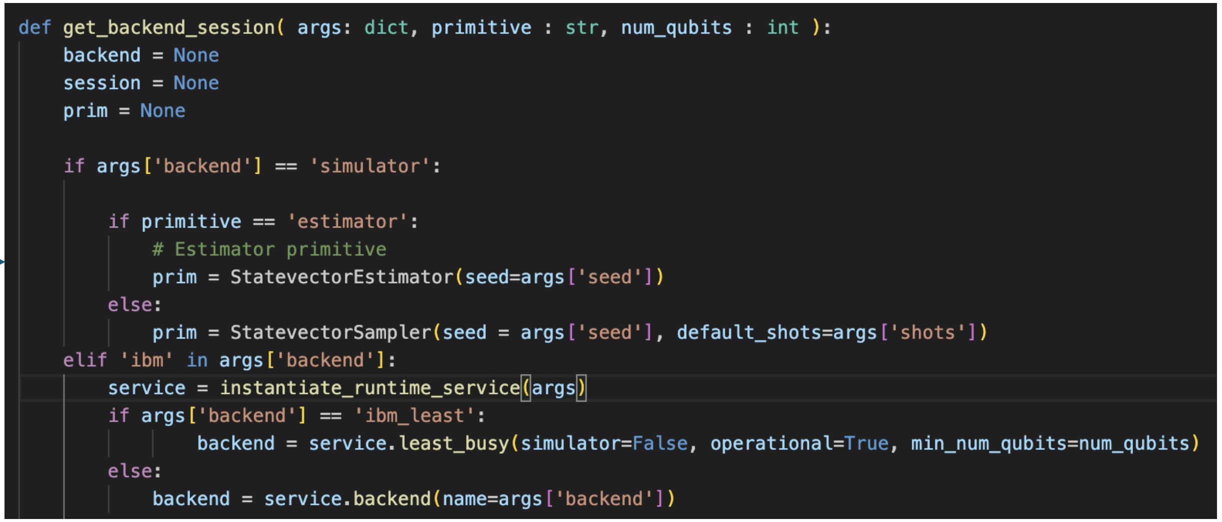
3) Quantum backend

config.yaml



```yaml
# choose a backend for the QML methods
# backend: 'ibm_cleveland'
# backend: 'ibm_least'
backend: 'simulator'

# IBM runtime credentials - they should be in
qiskit_json_path: '~/.qiskit/qiskit-ibm.json'
```

qml4omics/utils/**qutils.py**

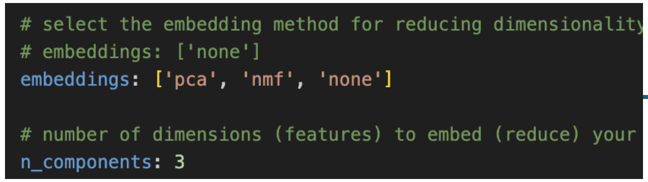


```python
def get_backend_session( args: dict, primitive : str, num_qubits : int ):
    backend = None
    session = None
    prim = None

    if args['backend'] == 'simulator':

        if primitive == 'estimator':
            # Estimator primitive
            prim = StatevectorEstimator(seed=args['seed'])
        else:
            prim = StatevectorSampler(seed = args['seed'], default_shots=args['shots'])
    elif 'ibm' in args['backend']:
        service = instantiate_runtime_service(args)
        if args['backend'] == 'ibm_least':
                backend = service.least_busy(simulator=False, operational=True, min_num_qubits=num_qubits)
        else:
            backend = service.backend(name=args['backend'])
```

So, what does the config file do?

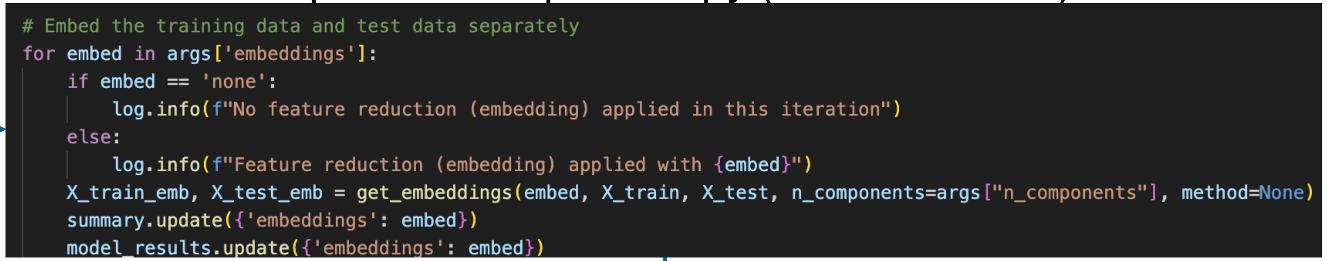
4) Embedding the data (reducing dimensions/features)

config.yaml

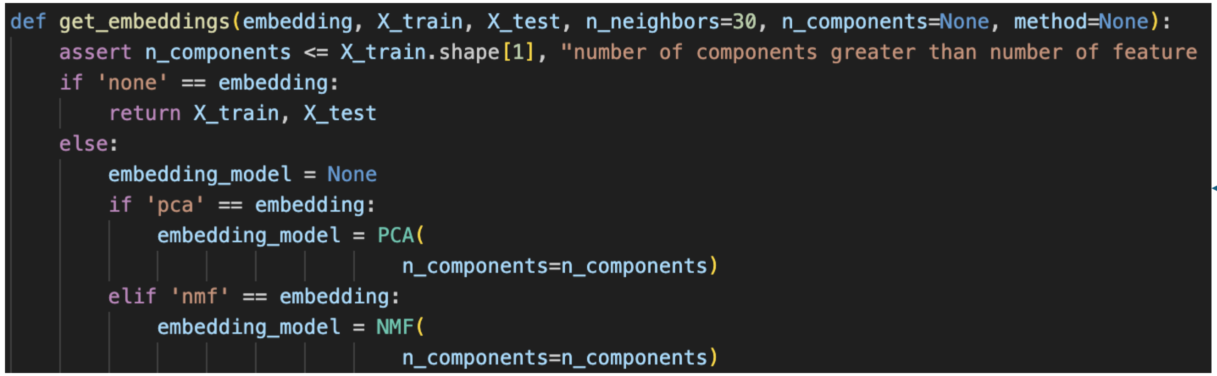


```yaml
# select the embedding method for reducing dimensionality
# embeddings: ['none']
embeddings: ['pca', 'nmf', 'none']

# number of dimensions (features) to embed (reduce) your
n_components: 3
```

qml4omics-profiler.py (main function)



```python
# Embed the training data and test data separately
for embed in args['embeddings']:
    if embed == 'none':
        log.info(f"No feature reduction (embedding) applied in this iteration")
    else:
        log.info(f"Feature reduction (embedding) applied with {embed}")
    X_train_emb, X_test_emb = get_embeddings(embed, X_train, X_test, n_components=args["n_components"], method=None)
    summary.update({'embeddings': embed})
    model_results.update({'embeddings': embed})
```



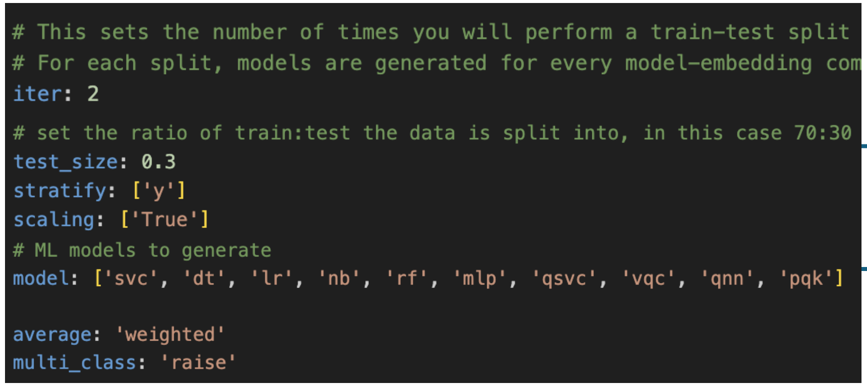
```python
def get_embeddings(embedding, X_train, X_test, n_neighbors=30, n_components=None, method=None):
    assert n_components <= X_train.shape[1], "number of components greater than number of feature
    if 'none' == embedding:
        return X_train, X_test
    else:
        embedding_model = None
        if 'pca' == embedding:
            embedding_model = PCA(
                                n_components=n_components)
        elif 'nmf' == embedding:
            embedding_model = NMF(
                                n_components=n_components)
```

qml4omics/embeddings/embed.py

So, what does the config file do?

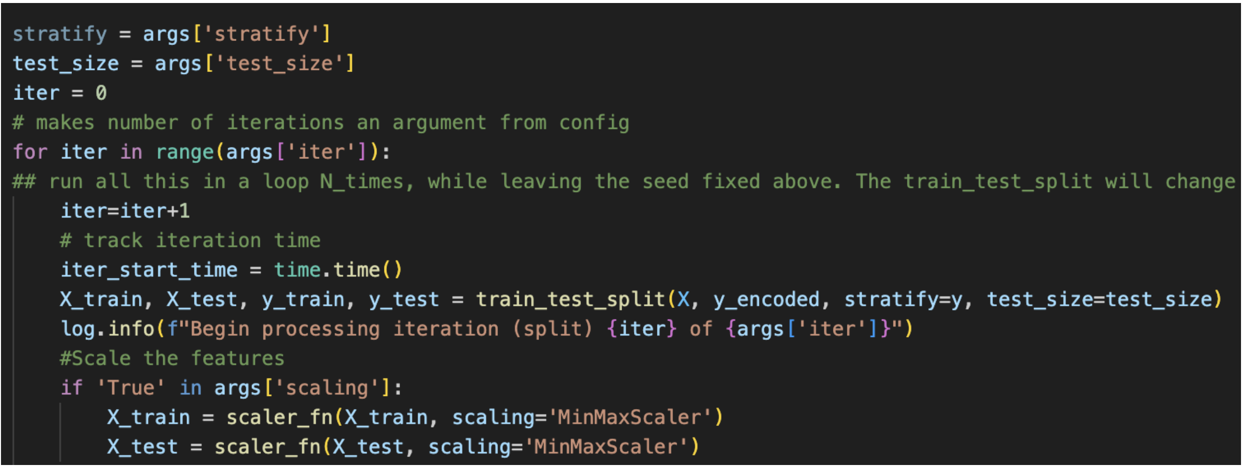
5) Splitting the data

config.yaml

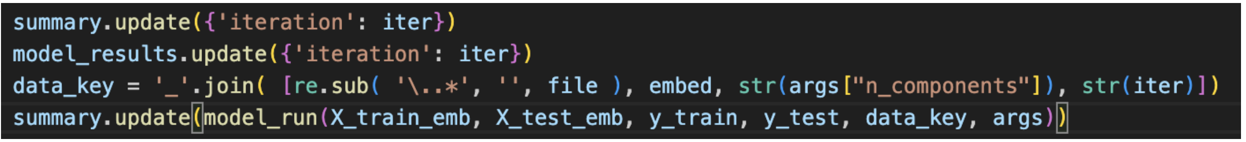


```
# This sets the number of times you will perform a train-test split
# For each split, models are generated for every model-embedding com
iter: 2

# set the ratio of train:test the data is split into, in this case 70:30
test_size: 0.3
stratify: ['y']
scaling: ['True']
# ML models to generate
model: ['svc', 'dt', 'lr', 'nb', 'rf', 'mlp', 'qsvc', 'vqc', 'qnn', 'pqk']

average: 'weighted'
multi_class: 'raise'
```

qml4omics-profiler.py (main function)



```
stratify = args['stratify']
test_size = args['test_size']
iter = 0
# makes number of iterations an argument from config
for iter in range(args['iter']):
## run all this in a loop N_times, while leaving the seed fixed above. The train_test_split will change
    iter=iter+1
    # track iteration time
    iter_start_time = time.time()
    X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, stratify=y, test_size=test_size)
    log.info(f"Begin processing iteration (split) {iter} of {args['iter']}")
    #Scale the features
    if 'True' in args['scaling']:
        X_train = scaler_fn(X_train, scaling='MinMaxScaler')
        X_test = scaler_fn(X_test, scaling='MinMaxScaler')
```

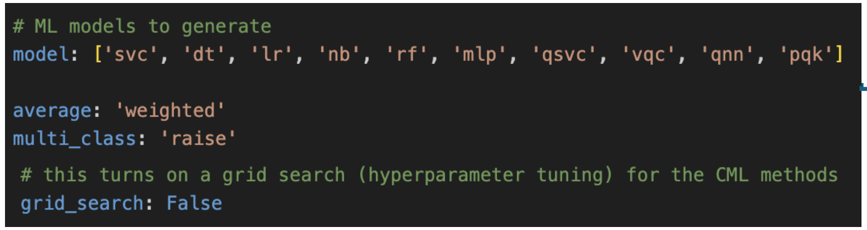


```
summary.update({'iteration': iter})
model_results.update({'iteration': iter})
data_key = '_'.join( [re.sub( '\..*', '', file ), embed, str(args["n_components"]), str(iter)])
summary.update(model_run(X_train_emb, X_test_emb, y_train, y_test, data_key, args))
```

So, what does the config file do?

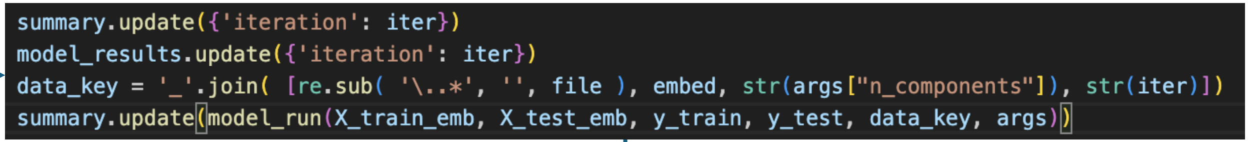
5) Run models (with and without grid search/hyperparameter tuning)

config.yaml

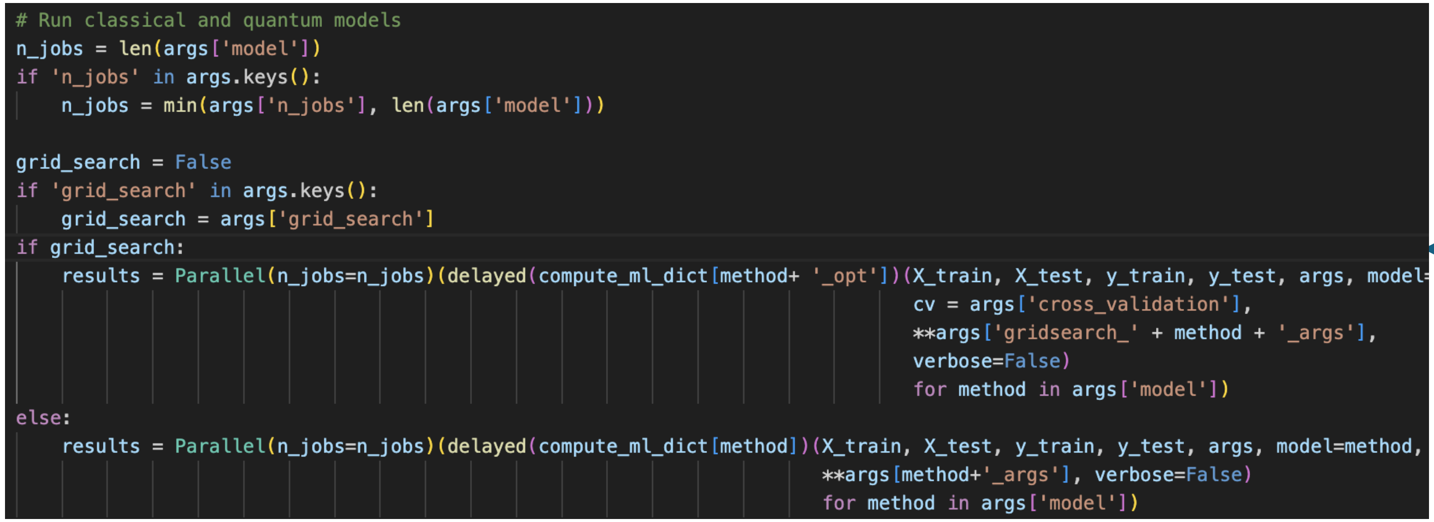


```yaml
# ML models to generate
model: ['svc', 'dt', 'lr', 'nb', 'rf', 'mlp', 'qsvc', 'vqc', 'qnn', 'pqk']

average: 'weighted'
multi_class: 'raise'
 # this turns on a grid search (hyperparameter tuning) for the CML methods
 grid_search: False
```

qml4omics-profiler.py (main function)



```python
summary.update({'iteration': iter})
model_results.update({'iteration': iter})
data_key = '_'.join( [re.sub( '\..*', '', file ), embed, str(args["n_components"]), str(iter)])
summary.update(model_run(X_train_emb, X_test_emb, y_train, y_test, data_key, args))
```



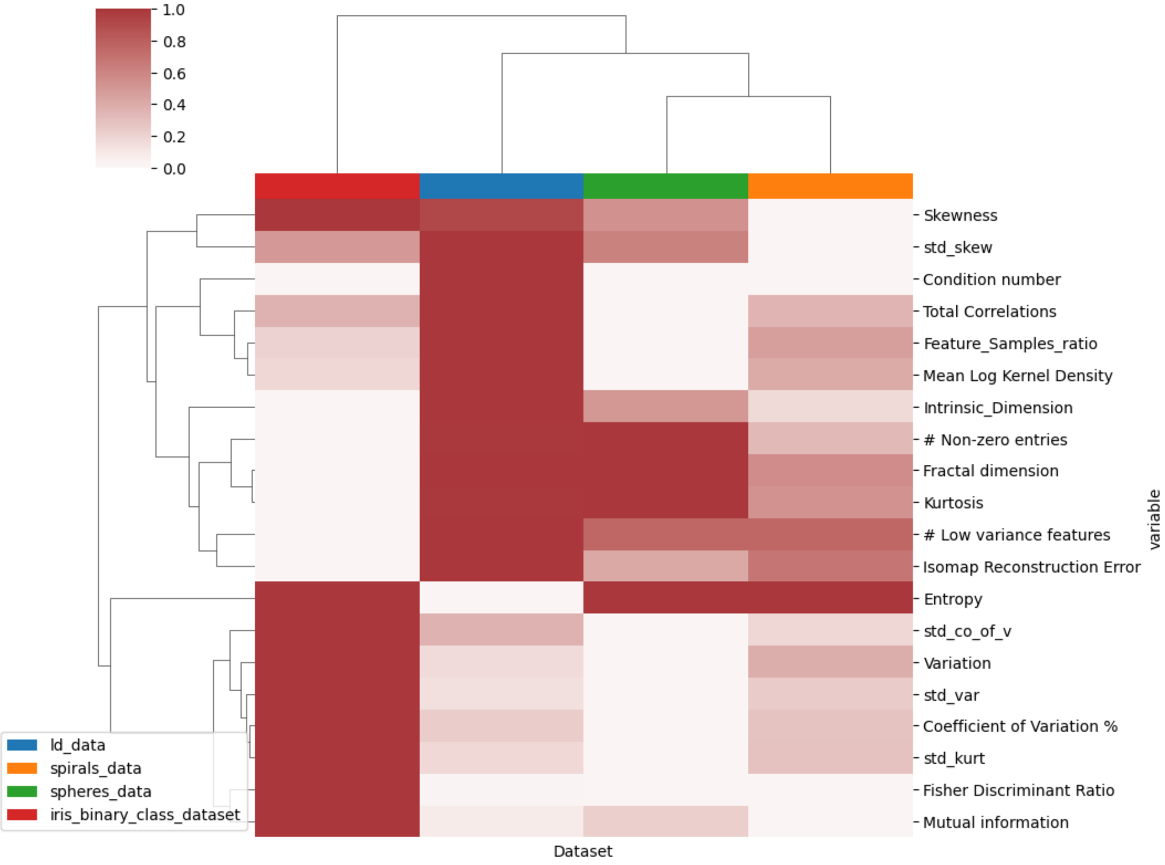
```python
# Run classical and quantum models
n_jobs = len(args['model'])
if 'n_jobs' in args.keys():
    n_jobs = min(args['n_jobs'], len(args['model']))

grid_search = False
if 'grid_search' in args.keys():
    grid_search = args['grid_search']
if grid_search:
    results = Parallel(n_jobs=n_jobs)(delayed(compute_ml_dict[method+ '_opt'])(X_train, X_test, y_train, y_test, args, model=
                                                                               cv = args['cross_validation'],
                                                                               **args['gridsearch_' + method + '_args'],
                                                                               verbose=False)
                                                                               for method in args['model'])
else:
    results = Parallel(n_jobs=n_jobs)(delayed(compute_ml_dict[method])(X_train, X_test, y_train, y_test, args, model=method,
                                                                       **args[method+'_args'], verbose=False)
                                                                       for method in args['model'])
```

qml4omics/evaluation/model_run.py

Understanding the Analyses

1) Hierarchical clustering heat maps

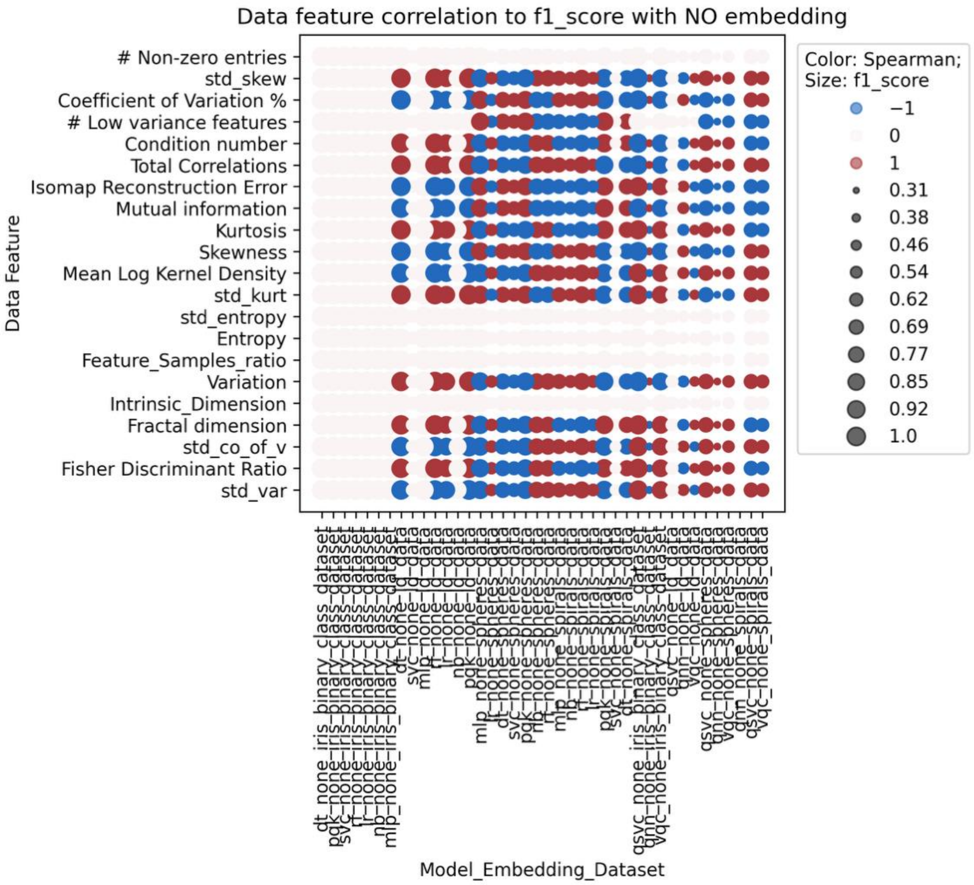
- What it’s doing here:
  - Complexity measure range is normalized between 0 and 1.
  - Euclidean distance is calculated between columns and rows, clustering together those with the shortest distance → similar intensities for complexity measures.
  - The dendogram branches create a pairing hierarchy.
  - Outlier has longest branch.
  - Helps answer: *Is there some structure or pattern in my data?*



Understanding the Analyses

1) Spearman Rank Correlations

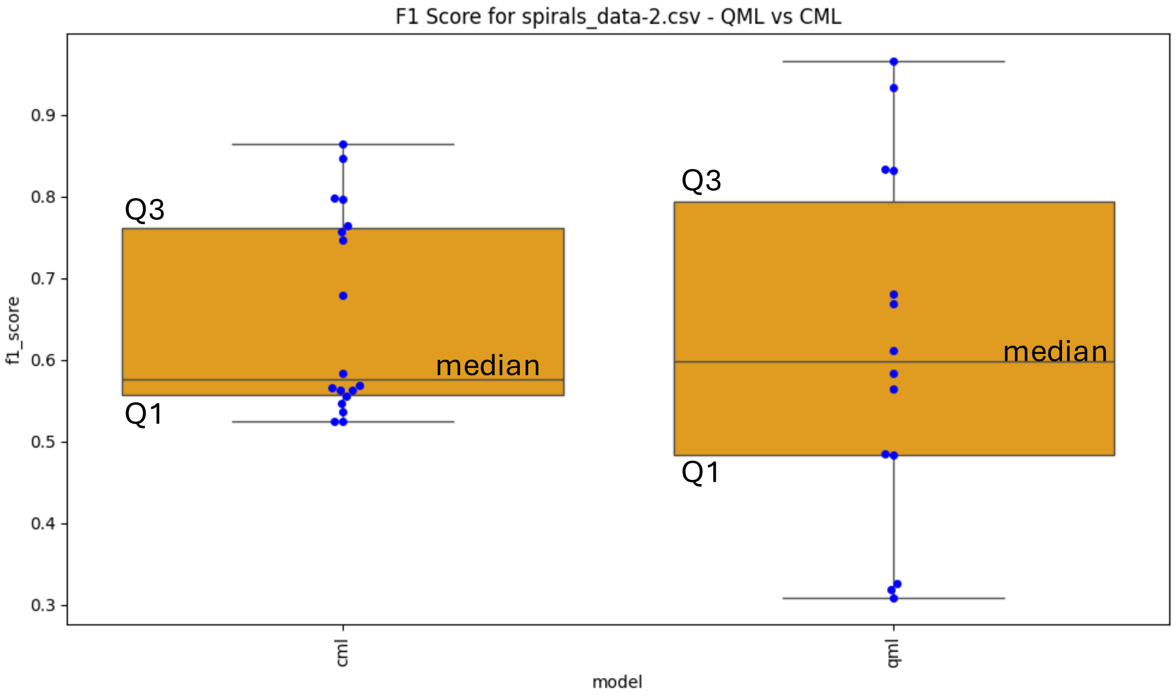
- What it’s doing here:
    - Correlates data complexity measure to model performance (F1-score)
    - Red = positive correlated
    - Blue = anti-correlated
    - Size of sphere = magnitude of F1-score
    - Helps answer: *What complexity measures influence your model score the most?*



Understanding the Analyses

1) Box-and-whisker plots

- What it’s doing here:
  - Plots distribution of median F1-scores per datasets, across all splits of data, per model.
  - Top and bottom of box = upper and lower quartiles (Q3 and Q1)
  - Whiskers denote range in F1- scores
  - Helps answer: *What is the locality, spread, and skewness groups in my data (F1-scores) based on their quartiles?*


F1 Score for spirals_data-2.csv - QML vs CML
f1_score
Q3
Q1
median
Q3
Q1
median
cml
qml
model