```python
# %% Load data into dataframes, set roles and create views
import getml

import cora.helpers as helpers

getml.engine.launch()
getml.engine.set_project("cora_visualization")

conn = getml.database.connect_mysql(
    host="db.relational-data.org",
    dbname="CORA",
    port=3306,
    user="guest",
    password="relational",
)

df_paper = getml.data.DataFrame.from_db(name="df_paper", table_name="paper", conn=conn)
df_cites = getml.data.DataFrame.from_db(name="df_cites", table_name="cites", conn=conn)
df_content = getml.data.DataFrame.from_db(name="df_content", table_name="content", conn=conn)

df_paper.set_role("paper_id", getml.data.roles.join_key)
df_paper.set_role("class_label", getml.data.roles.categorical)

df_cites.set_role(["cited_paper_id", "citing_paper_id"], getml.data.roles.join_key)

df_content.set_role("paper_id", getml.data.roles.join_key)
df_content.set_role("word_cited_id", getml.data.roles.categorical)

v_paper_multi_varget = getml.data.make_target_columns(df_paper, "class_label")

# %% Define the data model
import getml.data.relationship as rel
from getml.data.placeholder import Placeholder

dm = getml.data.DataModel(population=Placeholder(name="ph_population", roles=v_paper_multi_varget.roles))

dm.add(Placeholder(name="ph_cites", roles=df_cites.roles))
dm.add(Placeholder(name="ph_cites", roles=df_cites.roles))

dm.add(Placeholder(name="ph_content", roles=df_content.roles))
dm.add(Placeholder(name="ph_paper", roles=df_paper.roles))

dm["ph_population"].join(dm["ph_cites"][0], on=("paper_id", "cited_paper_id"))
dm["ph_cites"][0].join(dm["ph_content"], on=("citing_paper_id", "paper_id"))
dm["ph_cites"][0].join(dm["ph_paper"], on=("citing_paper_id", "paper_id"), relationship=rel.many_to_one)

dm["ph_population"].join(dm["ph_cites"][1], on=("paper_id", "citing_paper_id"))
dm["ph_cites"][1].join(dm["ph_content"], on=("cited_paper_id", "paper_id"))
dm["ph_cites"][1].join(dm["ph_paper"], on=("cited_paper_id", "paper_id"), relationship=rel.many_to_one)

dm["ph_population"].join(dm["ph_content"], on="paper_id")

# %% Define a split and create a container connecting the dataframes with the data model
split = getml.data.split.random(train=0.7, test=0.3, seed=0)
container = getml.data.Container(population=v_paper_multi_varget, split=split)
container.add(ph_cites=df_cites, ph_content=df_content, ph_paper=df_paper)
container.freeze()

# %% Define a pipeline and fit it to the data
fast_prop = getml.feature_learning.FastProp(
    loss_function=getml.feature_learning.loss_functions.CrossEntropyLoss,
    num_threads=10,
    aggregation=getml.feature_learning.aggregations.fastprop.Minimal,
    num_features=600,
)

pipe = getml.pipeline.Pipeline(
    tags=["fast_prop_mapping"],
    preprocessors=[getml.preprocessors.Mapping()],
    data_model=dm,
    feature_learners=[fast_prop],
    predictors=[getml.predictors.XGBoostClassifier(objective="binary:logistic")],
)

pipe.fit(container.train)
prob_vecs_1_vs_all_test = pipe.predict(container.test)

predicted_labels_test = helpers.probs_1_vs_all_to_label_via_argmax(
    prob_vecs_1_vs_all_test, class_labels=df_paper.class_label.unique()
)
gt_labels_test = df_paper[split == "test"].class_label.to_numpy()
accuracy = (gt_labels_test == predicted_labels_test).sum() / len(gt_labels_test)
```