Potential issues Python script for preprocessing, written exclusively **Corresponding dataflow DAG for** in preprocessing with native pandas and sklearn constructs instrumentation, extracted by *mlinspect* pipeline: # load input data sources, join to single table Data Source Data Source patients = pandas.read csv(...) histories = pandas.read csv(...) Join Aggregate Join might group by age group data = pandas.merge([patients, histories], on=['ssn']) on ssn change proportions # compute mean complications per age group, append as column Join on age group of groups in data complications = data.groupby('age group') .add(mean complications=('complications', 'mean')) **Declarative inspection** Project comp. Project mean. Column 'age group' data = data.merge(complications, on=['age group']) of preprocessing pipeline projected out, but # Target variable: people with frequent complications Project label required for fairness data['label'] = data['complications'] > mlinspect 1.2 \* data['mean complications'] **Project** PipelineInspector smoker, lastname, county, Selection might # Project data to subset of attributes, filter by counties .on pipeline('health.py') children, gender, income, label data = data[['smoker', 'last name', 'county', change proportions .no bias introduced for( 'num children', 'gender', 'income', 'label']] ['age group', 'gender']) of groups in data Filter data = data[data['county'].isin(counties of interest)] .no illegal features() county .no missing embeddings() # Define a nested feature encoding pipeline for the data Imputation might .verify() Split Split impute and encode = sklearn.Pipeline([ change proportions Training set Test set (sklearn.SimpleImputer(strategy='most frequent')), of groups in data (sklearn.OneHotEncoder())1) Project **Project** Project Project Project **Project** Project featurisation = sklearn.ColumnTransformer(transformers=[ n\_child. income gender smoker lastname county label (impute and encode, ['smoker', 'county', 'gender']), 'gender' as a feature (Word2VecTransformer(), 'last name') might be illegal! (sklearn.StandardScaler(), ['num children', 'income']]) Impute Impute Embed Scale Scale Impute # Define the training pipeline for the model smoker lastname n\_child. income county gender Embedding vectors neural net = sklearn.KerasClassifier(build fn=create model()) may not be available pipeline = sklearn.Pipeline([ ('features', featurisation), Ifor rare names! **Encode** Encode Encode ('learning algorithm', neural net)]) smoker county | gender # Train-test split, model training and evaluation train data, test data = train test split(data) Concatenate Learner model = pipeline.fit(train data, train data.label) **Neural Network** print(model.score(test data, test data.label))