

## Potential issues in preprocessing pipeline:

- 1 Join might change proportions of groups in data
- 2 Column 'age\_group' projected out, but required for fairness
- 3 Selection might change proportions of groups in data
- 4 Imputation might change proportions of groups in data
- 5 'gender' as a feature might be illegal!
- 6 Embedding vectors may not be available for rare names!

## Python script for preprocessing, written exclusively with native pandas and sklearn constructs

```
# load input data sources, join to single table
patients = pandas.read_csv(...)
histories = pandas.read_csv(...)
data = pandas.merge([patients, histories], on=['ssn'])

# compute mean complications per age group, append as column
complications = data.groupby('age_group')
    .agg(mean_complications=('complications', 'mean'))
data = data.merge(complications, on=['age_group'])

# Target variable: people with frequent complications
data['label'] = data['complications'] >
    1.2 * data['mean_complications']

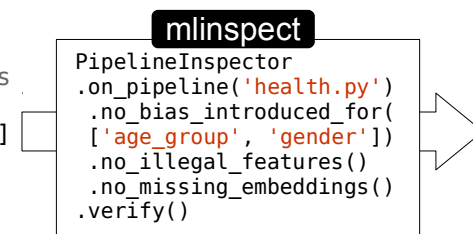
# Project data to subset of attributes, filter by counties
data = data[['smoker', 'last_name', 'county',
            'num_children', 'gender', 'income', 'label']]
data = data[data['county'].isin(counties_of_interest)]

# Define a nested feature encoding pipeline for the data
impute_and_encode = sklearn.Pipeline([
    (sklearn.SimpleImputer(strategy='most_frequent')),
    (sklearn.OneHotEncoder())])
featurisation = sklearn.ColumnTransformer(transformers=[
    (impute_and_encode, ['smoker', 'county', 'gender']),
    (Word2VecTransformer(), 'last_name')
    (sklearn.StandardScaler(), ['num_children', 'income'])])

# Define the training pipeline for the model
neural_net = sklearn.KerasClassifier(build_fn=create_model())
pipeline = sklearn.Pipeline([
    ('features', featurisation),
    ('learning_algorithm', neural_net)])

# Train-test split, model training and evaluation
train_data, test_data = train_test_split(data)
model = pipeline.fit(train_data, train_data.label)
print(model.score(test_data, test_data.label))
```

## Declarative inspection of preprocessing pipeline



## Corresponding dataflow DAG for instrumentation, extracted by *mlinspect*

