Lab #4: Variable Selection and Regularization

CS 109A, STAT 121A, AC 209A: Data Science

Fall 2016
Harvard

Today's lab: Problem 1

a) Visualize correlation in data

b) Select minimal subset of predictors

   – Exhaustive search

   – Step-wise forward search

Correlation Matrix

- Pearson correlation coefficient:
  - Measure of linear dependence between the predictors *i* and *j*

Covariance(X_i, X_j)

$$\rho_{ij} = \frac{\mathbf{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right]}{\sqrt{\mathbf{E}\left[(X_i - \mu_i)^2\right]}\sqrt{\mathbf{E}\left[(X_j - \mu_j)^2\right]}}$$

where $\mu_i = \mathbf{E}[X_i]$, $\mu_j = \mathbf{E}[X_j]$

Correlation Matrix

- Pearson correlation coefficient:
  - Measure of linear dependence between the predictors *i* and *j*

Covariance(X_i, X_j)

$$\rho_{ij} = \frac{\mathbf{E}\left[(X_i - \mu_i)(X_j - \mu_j)\right]}{\sqrt{\mathbf{E}\left[(X_i - \mu_i)^2\right]}\sqrt{\mathbf{E}\left[(X_j - \mu_j)^2\right]}}$$

where $\mu_i = \mathbf{E}[X_i]$, $\mu_j = \mathbf{E}[X_j]$

- What is the dimension of the correlation matrix?

Solve Part 1(a)

- Compute correlation matrix
  - `np.corrcoef(...)`

- Visualize correlation matrix:
  - `ax.pcolor(...)`

Variable Selection

Exhaustive Search

- For each size ‘k’:
  - Enumerate all subsets of size ‘k’
  - Fit regression model for each subset
  - Pick subset with maximum R^2
- Use BIC to choose best size, and output optimal subset for that size

Enumerating Subsets

- Enumerate all subsets of predictors {0, 1, 2 , 3}
  - Subsets of size 1: {0}, {1}, {2}, {3}
  - Subsets of size 2: {0, 1}, {0, 2}, {0, 3},
    {1, 2}, {1, 3}, {2, 3}
  - Subsets of size 3: {0, 1, 2}, {0, 1, 3},
    {0, 2, 3}, {1, 2, 3}
  - Subsets of size 4: {0, 1, 2 , 3}

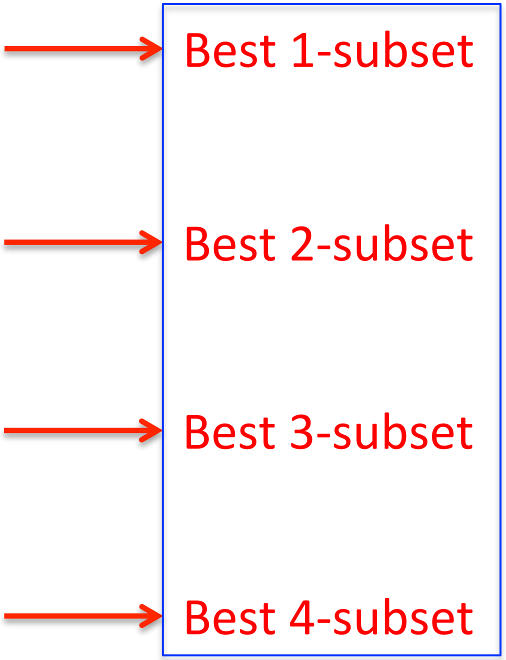
Enumerating Subsets

Best R^2 within each group

- Enumerate all subsets of predictors {0, 1, 2 , 3}
  - Subsets of size 1: {0}, {1}, {2}, {3} → Best 1-subset
  - Subsets of size 2: {0, 1}, {0, 2}, {0, 3}, {1, 2}, {1, 3}, {2, 3} → Best 2-subset
  - Subsets of size 3: {0, 1, 2}, {0, 1, 3}, {0, 2, 3}, {1, 2, 3} → Best 3-subset
  - Subsets of size 4: {0, 1, 2 , 3} → Best 4-subset

Enumerating Subsets

- Enumerate all subsets of predictors {0, 1, 2 , 3}
  - Subsets of size 1: {0}, {1}, {2}, {3}
  - Subsets of size 2: {0, 1}, {0, 2}, {0, 3}, {1, 2}, {1, 3}, {2, 3}
  - Subsets of size 3: {0, 1, 2}, {0, 1, 3}, {0, 2, 3}, {1, 2, 3}
  - Subsets of size 4: {0, 1, 2 , 3}



Best 1-subset

Best 2-subset

Best 3-subset

Best 4-subset

Choose subset with lowest BIC

Enumerating Subsets

- Generate all subsets of `set` of size `k`

```
subsets_k = itertools.combinations(set, k)
```

- Output is a list-like object

- Iterating through the generated subsets

```
for subset in subsets_k:
    …
```

Putting it together

```
# Outer loop: iterate over sizes 1 …. d
for k in range(d):
    # Enumerate subsets of size ‘k’
    subsets_k = itertools.combinations(predictors, k)
```

Putting it together

```
# Outer loop: iterate over sizes 1 …. d
for k in range(d):
    # Enumerate subsets of size ‘k’
    subsets_k = itertools.combinations(predictors, k)

    # Inner loop: iterate through subsets_k
    for subset in subsets_k :
        # Fit regression model using ‘subset’ and calculate R^2
        # Keep track of subset with highest R^2
        …
```

Putting it together

```
# Outer loop: iterate over sizes 1 …. d
for k in range(d):
    # Enumerate subsets of size 'k'
    subsets_k = itertools.combinations(predictors, k)
    # Inner loop: iterate through subsets_k
    for subset in subsets_k :
        # Fit regression model using 'subset' and calculate R^2
        # Keep track of subset with highest R^2
        …
```

Finds
k-sized subset
with best R^2

Putting it together

```
# Outer loop: iterate over sizes 1 …. d
for k in range(d):
    # Enumerate subsets of size 'k'
    subsets_k = itertools.combinations(predictors, k)
    # Inner loop: iterate through subsets_k
    for subset in subsets_k :
        # Fit regression model using 'subset' and calculate R^2
        # Keep track of subset with highest R^2
        …
    # Compute BIC of the subset you get from the inner loop
    # Compare with lowest BIC so far
```

Finds k-sized subset with best R^2

Solve Part 1(b)

- Implement exhaustive search
- Implement step-wise forward selection

Step-wise Forward Selection

- Start with empty set
- Repeat for every subset size 1, …, d:
  - For each predictor not chosen so far: add the predictor and fit a regression model
  - Find predictor that improves the R^2 the most
- Use BIC to choose best subset size

Current & Remaining Lists

- Outer loop: iterate over 1, …, *d*

current_predictors

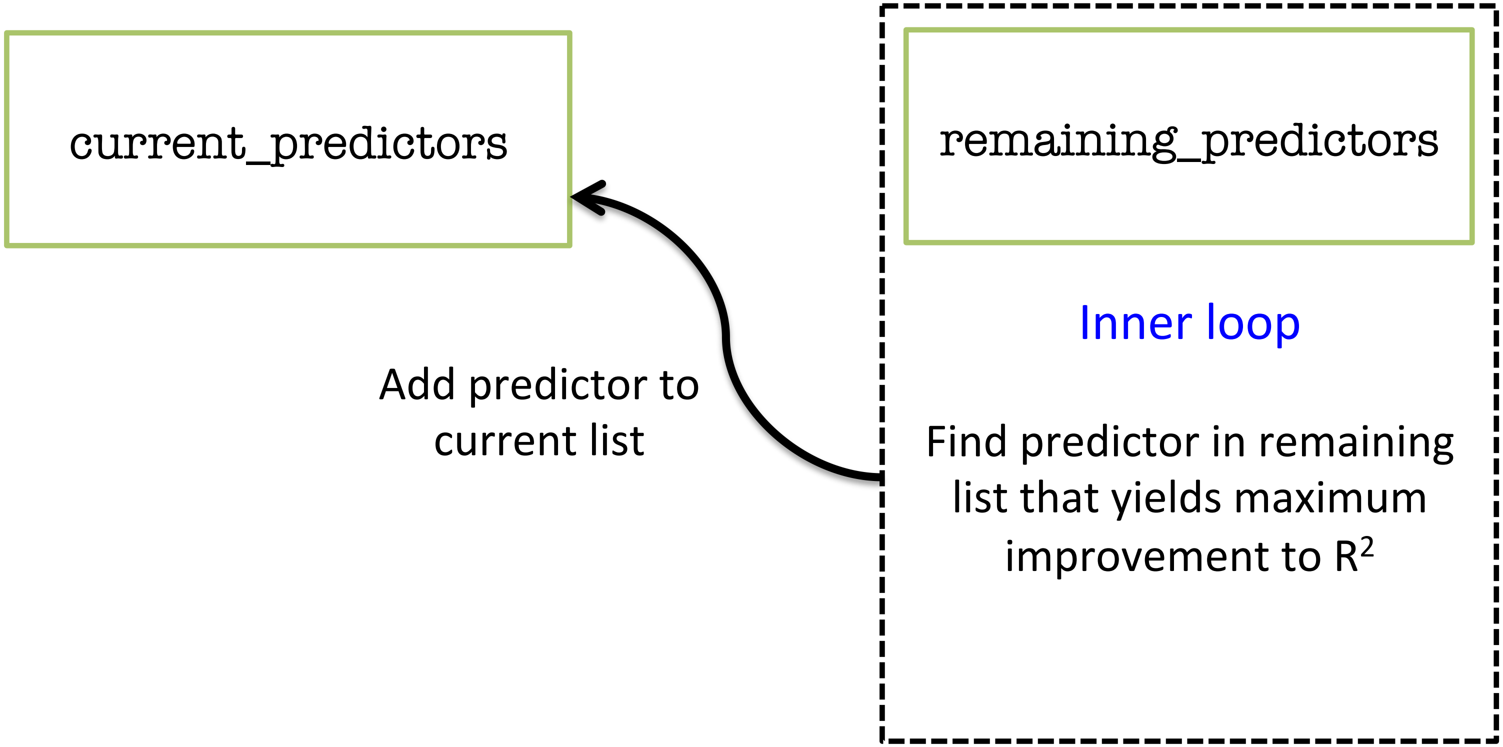
(initially: empty)

remaining_predictors

(initially: all predictors)

Current & Remaining Lists

- Outer loop: iterate over 1, …, *d*



current_predictors

remaining_predictors

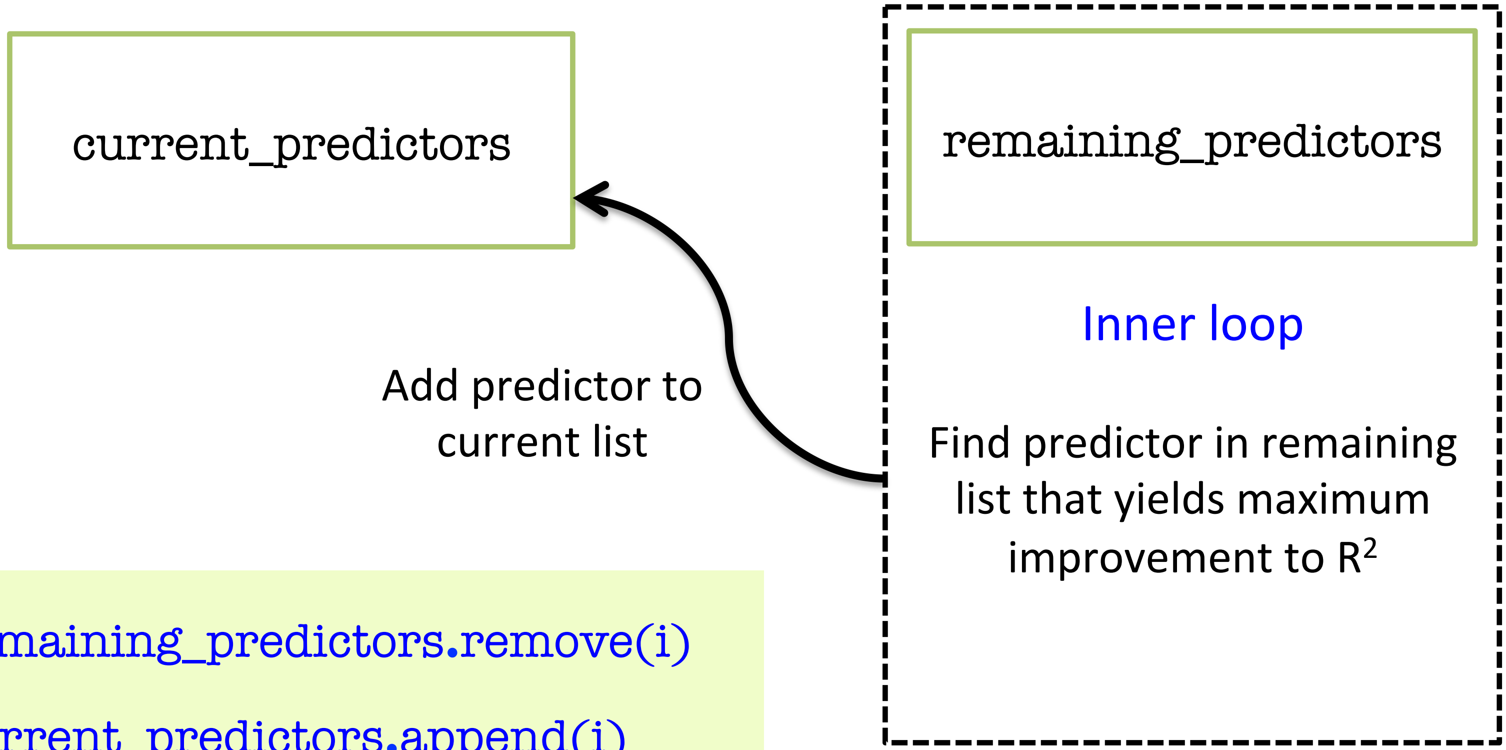
Inner loop

Find predictor in remaining list that yields maximum improvement to R^2

Add predictor to current list

Current & Remaining Lists

- Outer loop: iterate over 1, …, *d*


current_predictors
Add predictor to current list
remaining_predictors
Inner loop
Find predictor in remaining list that yields maximum improvement to R²

```
remaining_predictors.remove(i)
current_predictors.append(i)
```

Putting it together

```
current_predictors = []
remaining_predictors = range(d)

# Outer loop: iterate over sizes 1 .... d
for size in range(d):
```

Putting it together

```
current_predictors = []
remaining_predictors = range(d)

# Outer loop: iterate over sizes 1 …. d
for size in range(d):

    # Inner loop: iterate over remaining_predictors
    for i in remaining_predictors :
        # Make a copy of current_predictors, add ‘i’ to  the copied list
        # Fit regression model to the copied list, evaluate R^2
```

Putting it together

```
current_predictors = []
remaining_predictors = range(d)

# Outer loop: iterate over sizes 1 …. d
for size in range(d):
    # Inner loop: iterate over remaining_predictors
    for i in remaining_predictors :
        # Make a copy of current_predictors, add ‘i’ to  the copied list
        # Fit regression model to the copied list, evaluate R^2
```

Finds predictor that maximizes R^2 the most

Putting it together

```
current_predictors = []
remaining_predictors = range(d)

# Outer loop: iterate over sizes 1 .... d
for size in range(d):
    # Inner loop: iterate over remaining_predictors
    for i in remaining_predictors :
        # Make a copy of current_predictors, add 'i' to  the copied list
        # Fit regression model to the copied list, evaluate R^2

    # Add predictor you get to current_predictors
    # Remove the predictor from remaining_predictors
    # Compute BIC of current_predictors, and compare with best BIC so far
```

Finds predictor
that maximizes
R^2 the most

Solve Part 1(b)

- Implement exhaustive search
- Implement step-wise forward selection

Dealing with Categorical Predictors

- One-hot encoding: Binary encoding of categorical predictors
- If predictor *Z* takes *K* categories $\{c_1, \ldots, c_K\}$, replace it with *K* binary predictors $Z_1, .. Z_K$:
  - $Z_i = 1$ when *Z* takes value c_i and is 0 otherwise

One-hot Encoding in pandas

- How do you identify categorical attributes?
  - Look for data type (string or object)
  - Look for number of unique values (< 8?)

One-hot Encoding in pandas

- How do you identify categorical attributes?
  - Look for data type (string or object)
  - Look for number of unique values (< 8?)
- Transforming single predictor:
  - `pandas.get_dummies(predictor)`
  - Input is a single column as a df
  - Output is a df of multiple binary predictors

One-hot Encoding in pandas

- How do you identify categorical attributes?
  - Look for data type (string or object)
  - Look for number of unique values (< 8?)
- Transforming single predictor:
  - pandas.get_dummies(predictor)
  - Input is a single column as a df
  - Output is a df of multiple binary predictors
- Append new predictors to data frame: pandas.concat(...)

Putting it together

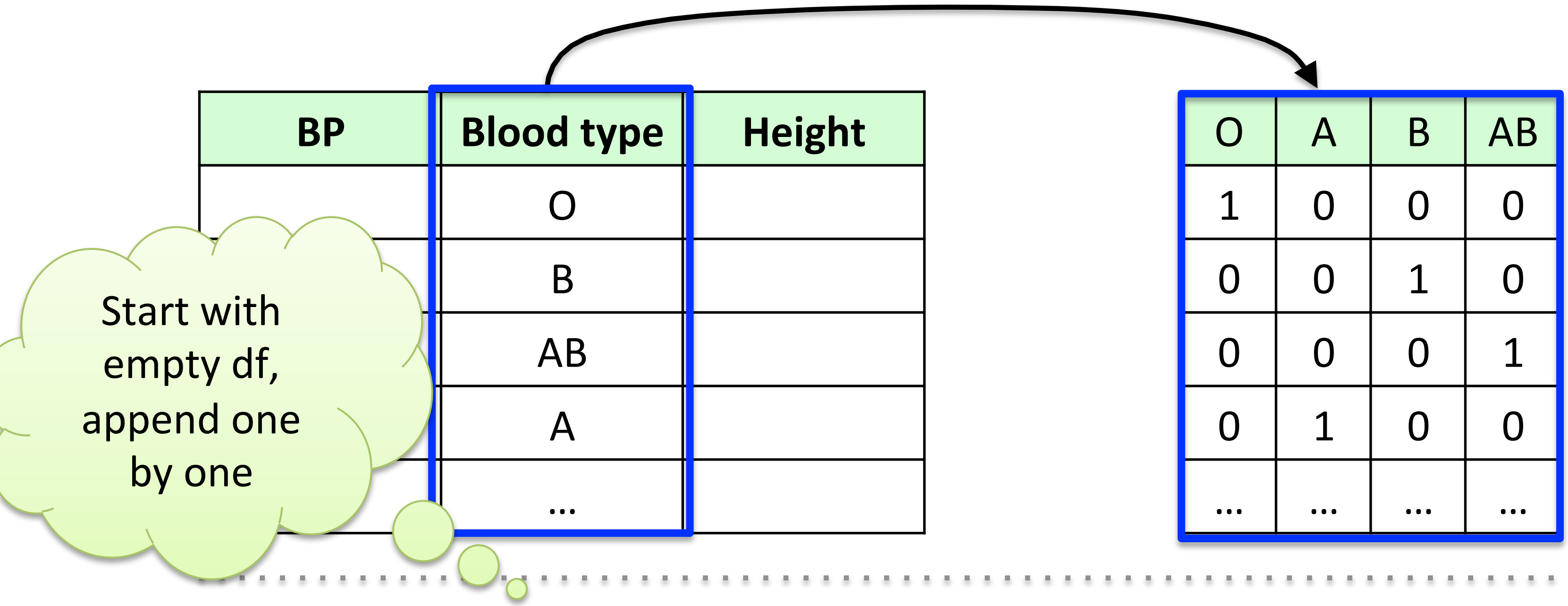
BP	Blood type	Height
	O	
	B	
	AB	
	A	
	…	

Putting it together

BP	Blood type	Height
	O	
	B	
	AB	
	A	
	…	

O	A	B	AB
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
…	…	…	…

Putting it together



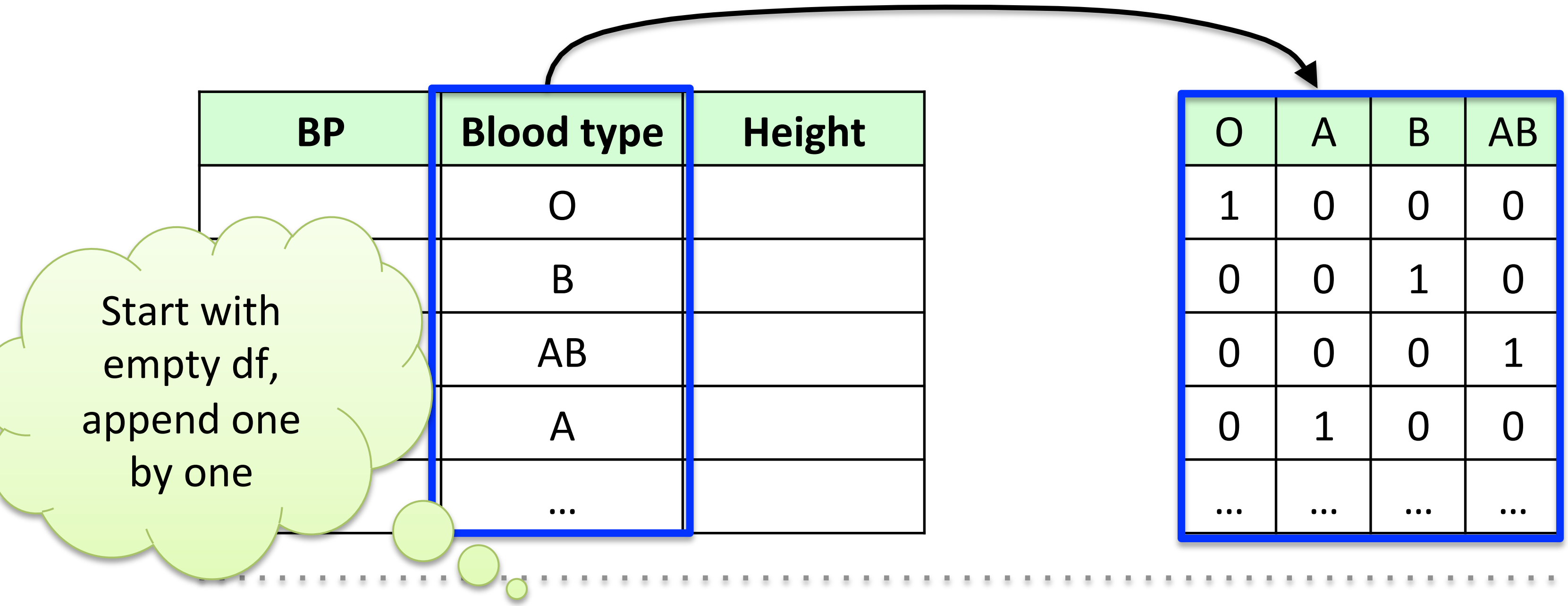
BP	Blood type	Height
	O	
	B	
	AB	
	A	
	…	

O	A	B	AB
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
…	…	…	…

Start with empty df, append one by one

BP

Putting it together



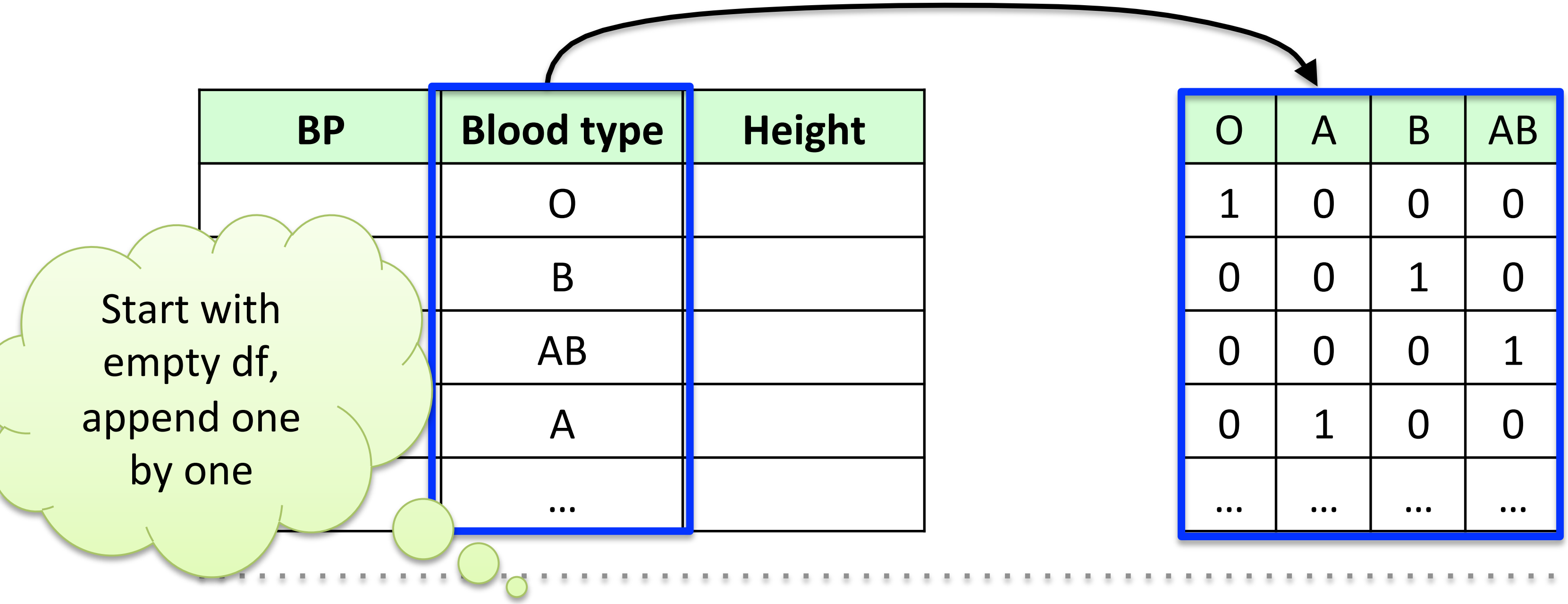
BP	Blood type	Height
	O	
	B	
	AB	
	A	
	…	

O	A	B	AB
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
…	…	…	…

Start with empty df, append one by one

BP	Blood type			
	O	A	B	AB
	1	0	0	0
	0	0	1	0
	0	0	0	1
	0	1	0	0
	…	…	…	…

Putting it together



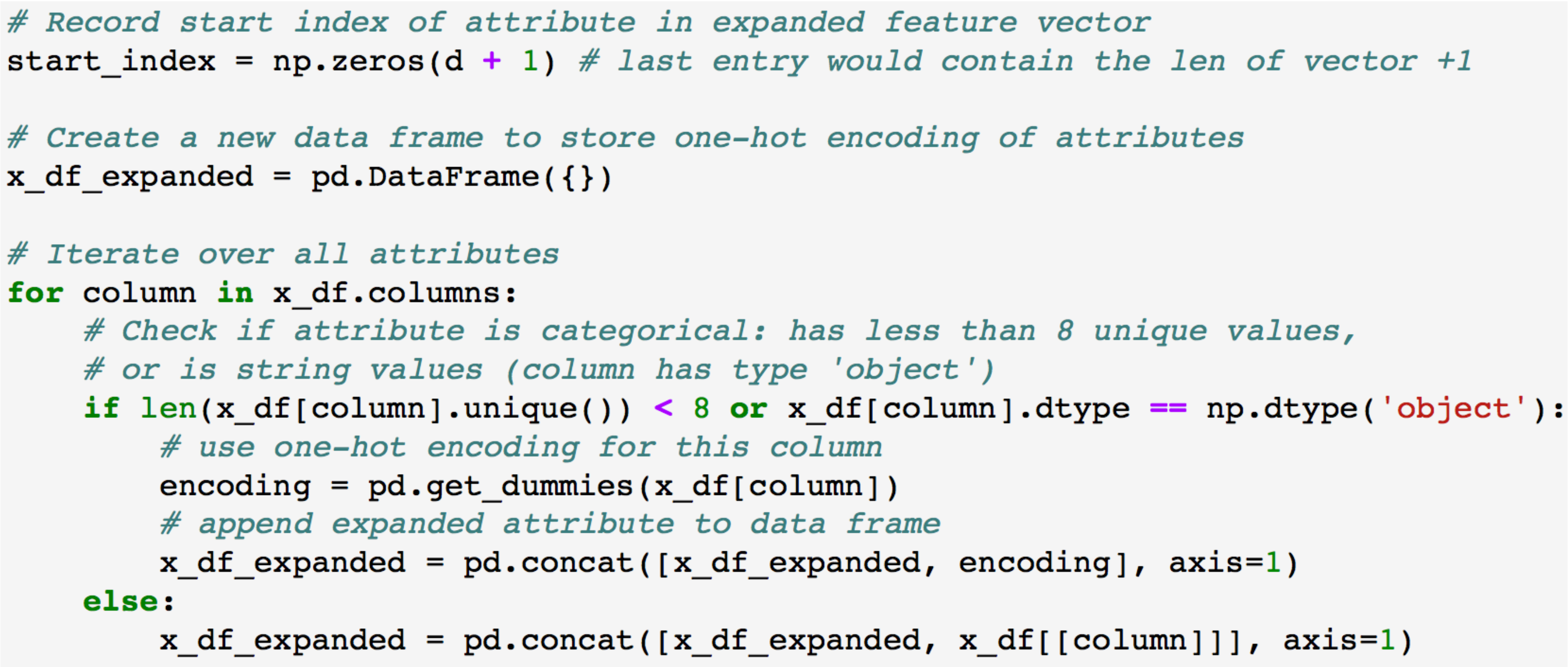
BP	Blood type	Height
	O	
	B	
	AB	
	A	
	…	

O	A	B	AB
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
…	…	…	…

Start with empty df, append one by one

BP	Blood type				Height
	O	A	B	AB	
	1	0	0	0	
	0	0	1	0	
	0	0	0	1	
	0	1	0	0	
	…	…	…	…	

Putting it together



```python
# Record start index of attribute in expanded feature vector
start_index = np.zeros(d + 1) # last entry would contain the len of vector +1

# Create a new data frame to store one-hot encoding of attributes
x_df_expanded = pd.DataFrame({})

# Iterate over all attributes
for column in x_df.columns:
    # Check if attribute is categorical: has less than 8 unique values,
    # or is string values (column has type 'object')
    if len(x_df[column].unique()) < 8 or x_df[column].dtype == np.dtype('object'):
        # use one-hot encoding for this column
        encoding = pd.get_dummies(x_df[column])
        # append expanded attribute to data frame
        x_df_expanded = pd.concat([x_df_expanded, encoding], axis=1)
    else:
        x_df_expanded = pd.concat([x_df_expanded, x_df[[column]]], axis=1)
```