To NumPy

Understand how DataFrames can be converted to 2-D NumPy arrays.

Chapter Goals:

- Learn how to convert a DataFrame to a NumPy matrix
- Write code to modify an MLB dataset and convert it to a NumPy matrix

A. Machine learning

The DataFrame object is great for storing a dataset and performing data analysis in Python. However, most machine learning frameworks (e.g. TensorFlow), work directly with NumPy data. Furthermore, the NumPy data used as input to machine learning models must solely contain quantitative values.

Therefore, to use a DataFrame's data with a machine learning model, we need to convert the DataFrame to a NumPy matrix of quantitative data. So even the categorical features of a DataFrame, such as gender and birthplace, must be converted to quantitative values.

B. Indicator features

When converting a DataFrame to a NumPy matrix of quantitative data, we need to find a way to modify the categorical features in the DataFrame.

The easiest way to do this is to convert each categorical feature into a set of *indicator features* for each of its categories. The indicator feature for a specific category represents whether or not a given data sample belongs to that category.

The code below shows a DataFrame with indicator features.

```
# predefined non-indicator DataFrame
print('{}\n'.format(df))

# predefined indicator Dataframe
```

```
print('{}\n'.format(indicator_df))
```

In the code above, the DataFrame `df` has a single categorical feature called `Color`. The corresponding indicator features for `Color` are shown in `indicator_df`.

Note that an indicator feature contains `1` when the row has that particular category, and `0` if the row does not.

C. Converting to indicators

In pandas, we convert each categorical feature of a DataFrame to indicator features with the `get_dummies` function. The function takes in a DataFrame as its required argument, and returns the DataFrame with each of its categorical features converted to indicator features.

The code below demonstrates how to use the `get_dummies` function.

```
# predefined df
print('{}\n'.format(df))

converted = pd.get_dummies(df)
print('{}\n'.format(converted.columns))

print('{}\n'.format(converted[['teamID_BOS',
                               'teamID_PIT']]))
print('{}\n'.format(converted[['lgID_AL',
                               'lgID_NL']]))
```

Note that the indicator features have the original categorical feature's label as a prefix. This makes it easy to see where each indicator feature originally came from.

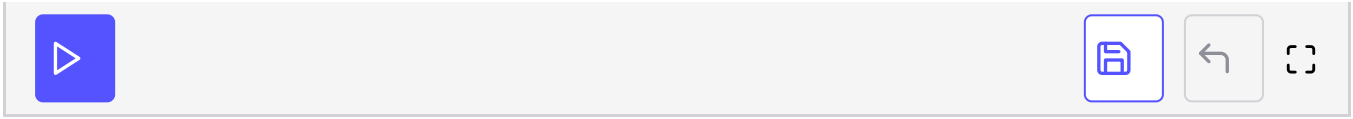
D. Converting to NumPy

After converting all the categorical features to indicator features, the DataFrame should have all quantitative data. We can then convert to a NumPy matrix using the `values` function.

The code below converts a DataFrame, `df` into a NumPy matrix.

```
# predefined indicator df
print('{}\n'.format(df))

n_matrix = df.values
print(repr(n_matrix))
```



The rows and columns of the output matrix correspond to the rows and columns of the same position in the DataFrame. In the code above, the first column of the NumPy matrix represents `HR`, the second column represents `teamID_BOS`, and the third column represents `teamID_PIT`.

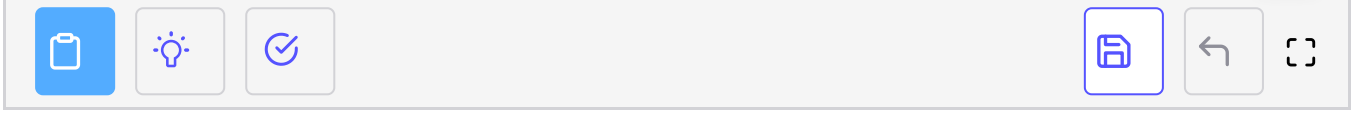
Time to Code!

The code exercise for this chapter will be to convert a DataFrame of MLB statistics (`df`) into a NumPy matrix.

We only want the data in `df` to be from the current century, so we need to first apply a filter.

Filter `df` for rows where `'yearID'` is at least `2000`, then reset `df` equal to the filtered output.

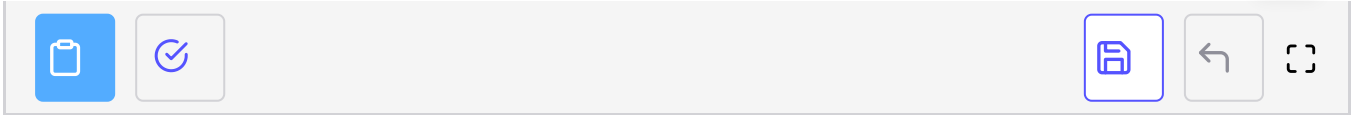
```
# CODE HERE
```



We also don't want any of the NaN values in our data. We can filter those out using the special `dropna` function.

Set `df` equal to `df.dropna` applied with no arguments.

```
# CODE HERE
```



Finally, we want to convert each categorical feature into a set of indicator features for each of its categories.

Then we can convert `df` into a NumPy matrix.

Set `df` equal to `pd.get_dummies` with `df` as the only argument.

Set `matrix` equal to `df.values`.

```
# CODE HERE
```