Indicator Columns

Learn about the indicator feature columns for the ML model's input layer.

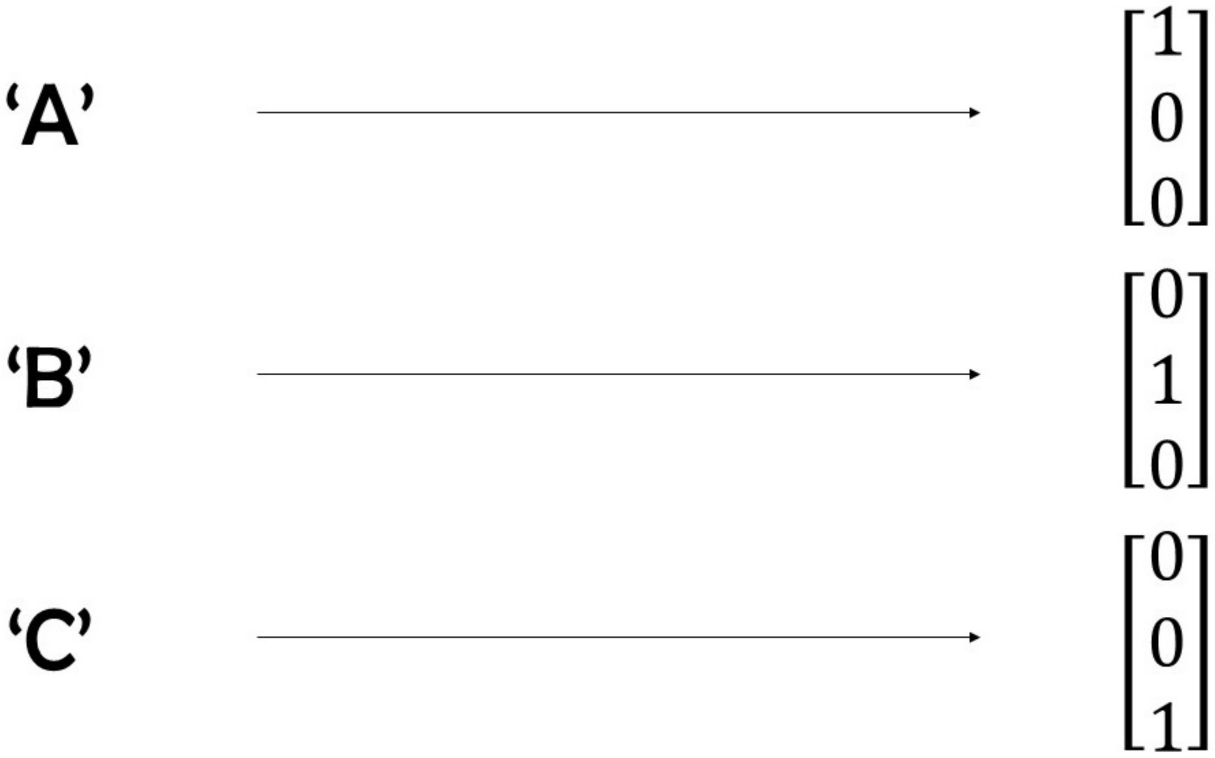
Chapter Goals:

- Process the indicator feature columns used for the machine learning model's input layer

A. One-hot indicators

The remaining non-numeric features in the dataset are the categorical features. Each of these features contains values that can be separated into a fixed number of distinct categories. For example, the `'IsHoliday'` feature contains the categories `0` and `1`, while the `'Type'` feature contains the categories `A`, `B`, and `C`.

These two features specifically (`'IsHoliday'` and `'Type'`) will be the indicator features for the dataset. Since indicator features are categorical, this means they will be represented by one-hot vectors when aggregated into the model's input layer.



Example one-hot vector representation for the 'Type' feature.

B. Categorical column base

When creating the indicator feature columns, we start off with a categorical column base. The categorical column specifies the distinct categories of the feature, as well as the feature's type. We'll use the `tf.feature_column.categorical_column_with_vocabulary_list` list to create the categorical columns.

```
import tensorflow as tf

type_col = tf.feature_column.categorical_column_with_vocabulary_list(
    'Type', ['A', 'B', 'C'], dtype=tf.string)
holiday_col = tf.feature_column.categorical_column_with_vocabulary_list(
    'IsHoliday', [0, 1], dtype=tf.int64)
```

Categorical columns for the 'IsHoliday' and 'Type' features.

We can then convert the categorical column bases into indicator feature columns with the `tf.feature_column.indicator_column` function.

```
import tensorflow as tf

type_feature_col = tf.feature_column.indicator_column(type_col)
holiday_feature_col = tf.feature_column.indicator_column(holiday_col)
```

Converting categorical columns to indicator feature columns.

Time to Code!

In this chapter you'll be creating the indicator feature columns for the dataset by completing the `add_indicator_columns` function. We've already filled the function with skeleton code that iterates through the indicator features in the dataset.

When creating the indicator feature columns, we need to specify the correct datatype. The `'Type'` feature column will have datatype `tf.string`, while the `'IsHoliday'` feature column will have datatype `tf.int64`.

`IsHoliday` feature column will have datatype `tf.int64`.

Set `dtype` equal to the correct datatype, depending on what `feature_name` is.

Each indicator feature column is built from a vocabulary list. The vocabulary list comes from the unique values of the feature in the `final_dataset` DataFrame.

Set `vocab_list` equal to the unique values in `final_dataset[feature_name]`, cast as a list.

Using the vocabulary list and datatype of the feature, we'll create the categorical column for the feature.

Set `vocab_col` equal to `tf.feature_column.categorical_column_with_vocabulary_list` with `feature_name` and `vocab_list` as the required arguments, as well as `dtype` for the `dtype` keyword argument.

After creating the categorical column, we can convert it into the required indicator feature.

Set `feature_col` equal to `tf.feature_column.indicator_column` applied to `vocab_col`. Then append `feature_col` to the end of the `feature_columns` list.

```
import tensorflow as tf

# Add the indicator feature columns to the list of feature columns
def add_indicator_columns(final_dataset, feature_columns):
    indicator_features = ['IsHoliday', 'Type']
    for feature_name in indicator_features:
        # CODE HERE
        pass
```