Sparse data structures

We can store sparse data using pandas which provides us with some efficient data structures. They need not be sparse in specific but also ‘0’. These data can also be some specific values like nan/ missing values. So by specifying these data object values can be omitted and compressed which saves data as these values are not actually stored.

SparseArray

For storing sparse values we use **.SparseArray** which is an extension array. It stores values distinct from the fill_value and is a 1D array.

We can convert the Sparse array back to regular (dense) ndarray using **numpy.asarray()**

SparseDtype

This property SparseArray.dtype can able to store two information

1. Dtype of non-sparse values
2. Scalar fill value

This can also be constructed passing only a dtype.

Sparse accessor

.sparse accessor, provides attributes and methods which are related to sparse data.
available only on data with SparseDtype

From scipy COO matrix we can create a Series with sparse data on the Seires class itself.
S.sparse.density is a specific property available on the .sparse accessor.

S.sparse.fill_value using this method we can always check the fill vales of the sparse dtype.

Sparse calculation

We get same arrays.SparseArray dtype as a result if numpy ufuncs are applied.

For example,
If we apply abs() function to a array with sparse values output we get will also be a array with sparse values.

Migrating

Initially, we used to have SparseSeries and SparseDataFrame classes to work on sparse data. After the advantage of using extension arrays, we are using a regular Series and DataFrame with sparse values in them.

Construction:

Previously we used .SparseDataFrame() to convert dataframe to sparse dataframe.
But now we convert the values of Series to sparse values using .Sparsearray and add each of them to the dataframe

Conversion:

To convert back sparse to dense we use a .sparse accessor .sparse.to_dense()
df.sparse.to_coo() Return the contents of the dataframe as a sparse SciPy matrix in COO format.

General differences:

Generally in a Dataframe, we have a mixture of sparse as well as dense columns but in SparseDataFrame all the columns were sparse. So, due to this assigning new columns to a general DataFrame with sparse values will not be sparse automatically. We need to ensure did we assigned sparse values are not.

Interaction with *scipy.sparse*

We can also convert a entire dataframe to a dataframe with columns of sparse values. For doing this first we convert dataframe to a sparse matrix using .sprse.csr_matrix()

Then the converted sparse_matrix is passed to **DataFrame.sparse.from_spmatrix()** in which columns list are passed are also passed as an argument to create a dataframe with sparse values. As a result if we check the dtypes of the final dataframe we can observe that all the columns are of sparse type.

Series.sparse.from_coo() is implemented for creating a **Series** with sparse values from a scipy.sparse.coo_matrix (sparse matrix in COO format)

We can also represent a series to a 2d array by mentioning multiindex levels of rows and columns. By passing an argument sort_label column and row lables can also be sorted in the final sparse representation.