Reducing Pandas Memory Usage

Pandas Datatyes

Pandas have some default data types that are assigned to the data frame when loaded.

1. **Int64**
2. **float64**
3. **object**

Ranges

Int8	-128 to 127
int16	-32768 to 32767
int32	-2 Billoin to 2 Billion
int64	9 Quintillion to 9 Quintillion

- **int64 consumes 8 times the memory as int8**

Solution

Increase in number of bits leads to increase in storage space. This applies to all data-types

We can downcast the variables

- **downcasting float64 to float16 would cut down the memory usage by 1/4th**

Let us see the implementation in Python

Loading the dataset

```
%%time
df = pd.read_csv('application_data.csv')

CPU times: user 2.68 s,
sys: 743 ms,
total: 3.43 s
Wall time: 3.53 s
```

Getting Info

```
print(df.info())

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
None
```

Observations

After observing the dataset I can say that there is no value greater than **2 billion**

- Downcasting **int64 to int32** is a good idea
- Downcasting the **flot64 to float 32** conveys enough accuracy

Downcasting

```
float_cols = [col for col in df if df[col].dtype=='float64']
int_cols = [col for col in df if df[col].dtype == 'int64']
df[float_cols] = df[float_cols].astype(np.float32)
df[int_cols] = df[int_cols].astype(np.int32)
```

Memory Usage

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 307511 entries, 0 to 307510
Columns: 122 entries, SK_ID_CURR to AMT_REQ_CREDIT_BUREAU_YEAR
dtypes: float32(65), int32(41), object(16)
memory usage: 161.9+ MB
```

Comparison

Before

```
dtypes: float64(65), int64(41), object(16)
memory usage: 286.2+ MB
```

After

```
dtypes: float32(65), int32(41), object(16)
memory usage: 161.9+ MB
```

- You can see more difference if you downcast to **int16 or float16**

Categorical

- If you have a categorical column like **gender, country, blood type, etc.** Then you can typecast that column to **'category', Reduces size.**

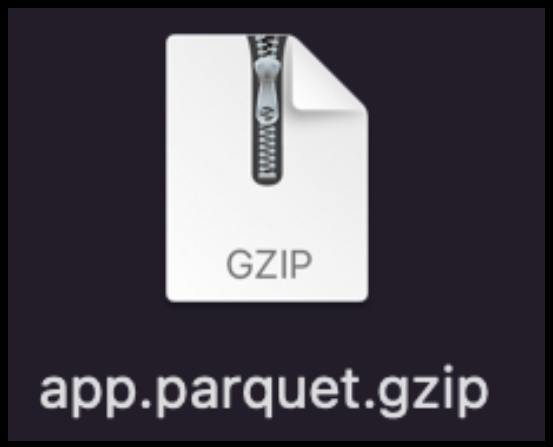
```
dataframe['column_name'].astype('category')
```

Bonus Step

Export the file in **parquet.gzip** format

- **This will help load the data faster**

```
df.to_parquet(
        "app.parquet.gzip",
        compression='gzip'  )
```



Reloading data

```
%%time
df = pd.read_parquet('app.parquet.gzip')
CPU times: user 867 ms,
sys: 149 ms,
total: 1.02 s
Wall time: 1.05 s
```

- At first the time taken was 3.53 seconds and now only **1.05** , bigger difference can be seen on **large datasets**

Thankyou for following this far

If you like my work follow me

Sadanand Ghule