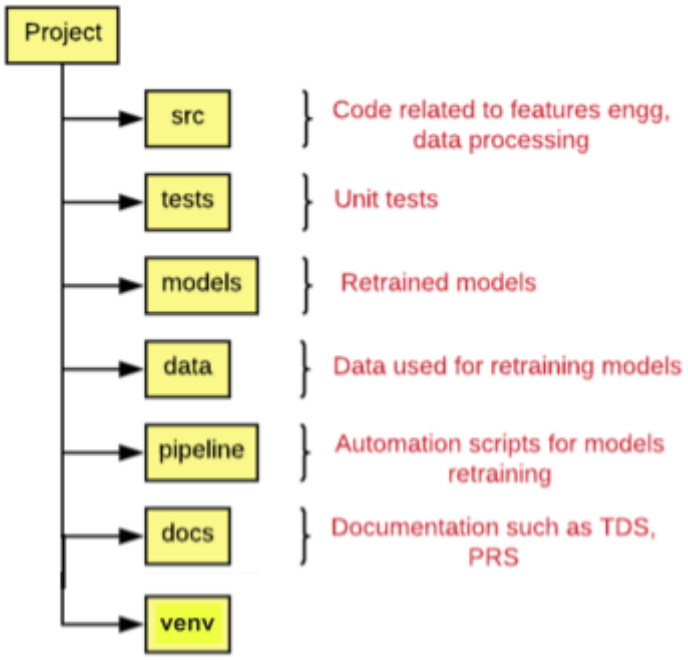
20. Pandas - Part 3

October 28, 2022

1 Folder/Directory Structure of a Data Science Project



The following are the details of the above-mentioned folder structure:

- **`Project`**: Name of the project.
- **`src`**: The folder that consists of the source code(Juputer Notebooks, python scripts, etc.) related to data gathering, data preparation, feature extraction, ML/DL model training, etc.
- **`tests`**: The folder that consists of the code representing unit tests for code maintained with the src folder.
- **`models`**: The folder that consists of files representing trained/retrained models as part of build jobs, etc. The model names can be appropriately set as projectname_date_time or project_build_id (in case the model is created as part of build jobs). Another approach is to store the model files in a separate storage such as AWS S3, Google Cloud Storage, or any other form of storage.

- **data**: The folder consists of data used for model training/retraining. The data could also be stored in a separate storage system (database).
- **pipeline**: The folder consists of code that's used for retraining and testing the model in an automated manner. These could be docker containers related - code, scripts, workflow related code, etc.
- **docs**: The folder that consists of code related to the Product Requirement Specifications (PRS), Technical Design Specifications (TDS), etc.
- **venv**: The folder is for python virtual environment for the project.

[ ]:
```
import pandas as pd
```

2 When reading from a file, how to read in only a subset of the columns?

[ ]:
```
df = pd.read_csv("../data/drinks.csv")
```

[ ]:
```
df.columns
```

[ ]:
```
df.head()
```

[ ]:
```
df.shape
```

[ ]:
```
# specify which columns to include by name
df = pd.read_csv('../data/drinks.csv', usecols=['country', 'continent'])
```

[ ]:
```
df.columns
```

[ ]:
```
df.head()
```

[ ]:
```
# or equivalently, specify columns by position
ufo_df = pd.read_csv('../data/ufo.csv', usecols=[0, 4])
```

[ ]:
```
ufo_df.columns
```

3 When reading from a file, how to read only a subset of the rows?

[ ]:
```
# specify how many rows to read
df = pd.read_csv('../data/drinks.csv', nrows=3)
```

[ ]:
```
df
```

4 How to iterate through a Series/Column?

[ ]:
```
df.head()
```

[ ]:
```
# Series are directly iterable (like a list)
for c in df.country:
    print(c)
```

5 How to iterate through a DataFrame?

[ ]:
```
# various methods are available to iterate through a DataFrame
for index, row in df.iterrows():
    print(index, row.country, row.continent)
```

6 How to know whether I should pass an argument as a string or a list?

[ ]:
```
df = pd.read_csv("../data/drinks.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
# describe all of the numeric columns
df.describe()
```

[ ]:
```
# pass the string 'all' to describe all columns
df.describe(include='all')
```

[ ]:
```
# pass a list of data types to only describe certain types
df.describe(include=['object', 'float64'])
```

[ ]:
```
# pass a list even if you only want to describe a single data type
df.describe(include=['object'])
```

7 How to use the "axis" parameter in pandas?

[ ]:
```
df.head()
```

[ ]:
```
# drop a column (temporarily)
df.drop('continent', axis=1).head()
```

[ ]:
```
# drop a row (temporarily)
df.drop(2, axis=0)
```

When **referring to rows or columns** with the axis parameter:

- **axis 0** refers to rows
- **axis 1** refers to columns

```
[ ]: # calculate the mean of each numeric column
df.mean(numeric_only=True)
```

```
[ ]: # or equivalently, specify the axis explicitly
df.mean(axis=0)
```

```
[ ]: # calculate the mean of each row
df.mean(axis=1).head()
```

When performing a **mathematical operation** with the axis parameter:

- **axis 0** means the operation should "move down" the row axis
- **axis 1** means the operation should "move across" the column axis

```
[ ]: # 'index' is an alias for axis 0
df.mean(axis='index')
```

```
[ ]: # 'columns' is an alias for axis 1
df.mean(axis='columns').head()
```

8 How to use string methods in pandas?

```
[ ]: df.head()
```

```
[ ]: # normal way to access string methods in Python
'hello'.upper()
```

```
[ ]: # string methods for pandas Series are accessed via 'str'
df.country = df.country.str.upper()
```

```
[ ]: df.continent.unique()
```

```
[ ]: # string method 'contains' checks for a substring and returns a boolean Series
df.continent.str.contains('America')
```

```
[ ]: # use the boolean Series to filter the DataFrame
df[df.continent.str.contains('America')]
```

9 How to change the data type of a Series?

```
[ ]: df.head()
```

```
[ ]: # examine the data type of each Series
     df.dtypes
```

```
[ ]: df.beer_servings.astype(float)
```

```
[ ]: # change the data type of an existing Series
     df['beer_servings'] = df.beer_servings.astype(float)
```

```
[ ]: df.dtypes
```

```
[ ]: # alternatively, change the data type of a Series while reading in a file
     df = pd.read_csv('../data/drinks.csv', dtype={'beer_servings':float})
```

```
[ ]: df.dtypes
```

```
[ ]: # string method 'contains' checks for a substring and returns a boolean Series
     df.continent.str.contains('America')
```

```
[ ]: # convert a boolean Series to an integer (False = 0, True = 1)
     df.continent.str.contains('America').astype(int)
```

10 groupby

```
[ ]: df.head()
```

```
[ ]: # calculate the mean beer servings across the entire dataset
     df.beer_servings.mean()
```

```
[ ]: df.continent=='Africa'
```

```
[ ]: df[df.continent=='Africa'].beer_servings
```

```
[ ]: # calculate the mean beer servings just for countries in Africa
     df[df.continent=='Africa'].beer_servings.mean()
```

```
[ ]: # calculate the mean beer servings for each continent
     df.groupby('continent').beer_servings.mean()
```

```
[ ]: # other aggregation functions (such as 'max') can also be used with groupby
     df.groupby('continent').beer_servings.max()
```

```
[ ]: # multiple aggregation functions can be applied simultaneously
     df.groupby('continent').beer_servings.agg(['count', 'mean', 'min', 'max'])
```

[ ]:
```
# specifying a column to which the aggregation function should be applied is␣
  ↪not required
df.groupby('continent').mean()
```

[ ]:
```
help(pd.DataFrame.plot)
```

[ ]:
```
# side-by-side bar plot of the DataFrame directly above
df.groupby('continent').mean().plot(kind='bar')
```