Data Exploration

Lets use a real life example to explore the useful data analysis features of pandas. For this purpose, we are going to use the Bike Shairing dataset that is made available on the UCI website. Using the various measures those are captured as part of this dataset, lets explore how pandas powerful capabilities can be utilized for the data wrangling and exploration.

Dataset and more information can be found at following URL -
https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset
(https://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset)
Please take some moments to go through the description of the dataset and its features which will be helpful for doing further data analysis.

The Basic Requirements

- Reading Data From CSV
- Formatting, cleaning and filtering Data Frames
- Group-by and Concat / Merge
- Writing Data to CSV

In [ ]:

```
import pandas as pd
import numpy as np
```

Step1 - Reading Data From CSV

As the data is stored in csv format, use read_csv function of pandas to read the data in pandas structure named DataFrame.

In [ ]:

```
bikes = pd.read_csv("bike_shairing_hourly.csv")
```

Data Viewing

Lets explore the first and last few rows of dataframe.

In [ ]:

```
bikes.head(2)
```

In [ ]:

```
bikes.tail(2)
```

Lets find out number of rows and columns of data set.

In [ ]:

```
bikes.shape
```

Step 2 - Formatting, cleaning and filtering Data Frames

Lets check how many data values are present in each column.

In [ ]:

```
bikes.info()
```

As all the features contains same number of data values, there are no missing values in dataset.

Properties of data

Lets check some properties of dataset :

In [ ]:

```
bikes.columns
```

18 features are present in dataset, most of which looks integers in nature.

In [ ]:

```
bikes.dtypes
```

dteday is datetime field so might need some transformation afterwards.

Unique feature values

Lets explore the unique values present in each feature. These unique values can give us some hints while doing grouping of the data.

In [ ]:

```
bikes.dteday.unique() # two years dates stored from 1 Jan 2011 to 31 Dec 2012
```

In [ ]:

```
bikes.season.unique()  # four seasons
```

Same thing can be done using [] operator

In [ ]:

```
bikes["season"].unique()  # four seasons
```

In [ ]:

```
bikes.yr.unique() # two values - 0 for year 2011 and 1 for year 2012
```

In [ ]:

```
bikes.mnth.unique() # 12 months data stored
```

In [ ]:

```
bikes.hr.unique()  # for each day of month, 24 hours data stored
```

In [ ]:

```
bikes.holiday.unique() #0- no holidy , 1 holiday
```

In [ ]:

```
bikes.weekday.unique() # each weekday given a number
```

In [ ]:

```
bikes.workingday.unique()  # workingday 1 otherwise 0
```

In [ ]:

```
bikes.weathersit.unique() #weathersit  has four values - 1 , 2, 3, 4
```

temp, atemp, hum, windspeed are real numbers so will not interested in unique values for it.

casual, registered and cnt are aggregated user counts for each day, so will not be interested in unique values for it.

Missing values identification

Missing values, null values, NaN needs to be identified.

In [ ]:

```
bikes.isnull().sum()
```

In [ ]:

```
bikes.isna().sum()
```

Individual column can also be chcked for the null values.

In [ ]:

```
bikes.dteday.isnull().sum()
```

None of the feature contains missing values or NaN.

If null values are present in the dataframe, following code needs to be executed in order to get rid of them.

In [ ]:

```
bikes = bikes.dropna()
#or inplace replacment in data frame
bikes.dropna(inplace=True)
bikes.count()
```

Null values can be removed from individual columns as well.

In [ ]:

```
bikes.yr.dropna(inplace=True)
bikes.yr.count()
```

Or missing values can be imputed with some other default values.

In [ ]:

```
bikes.mnth.fillna(0)  #Replace nulls with 0
bikes.mnth.unique()
```

Duplicate records identification

Duplicate records needs to be identified and removed.

In [ ]:

```
bikes.duplicated().sum()
```

There are no duplicate records in the data frame. Same duplicate check can be done on a column as well.

In [ ]:

```
bikes.instant.duplicated().sum()
```

If duplicate records are present, then following code can be used to remove them.

In [ ]:

```
bikes.drop_duplicates(subset="instant", keep="first", inplace=True)
bikes.count()
```

Data transformation

The datetime column "dte" is captured as object but as it contains date time in it, a lot of interesting things can be dervied from it and added to the existing data frame for further analysis.

In [ ]:

```
bikes.dteday.head()
```

Lets convert "dteday" to datetime.

In [ ]:

```
dte = pd.to_datetime(bikes.dteday)
type(dte)
```

Feature Extraction

Now extract year, month and day from "dte" series.

In [ ]:

```
year = dte.dt.year
print(year.head(2))
print(year.tail(2))
```

Add year series into the data frame.

In [ ]:

```
bikes["year"] = year
bikes.year.count()
```

Same way month, day , weekday features can be extracted from it and added to the dataframe. But our data frame already contains that information so we will not bother about it.

In [ ]:

```
months = dte.dt.month
days = dte.dt.day
day_of_weeks = dte.dt.dayofweek
day_of_year = dte.dt.dayofyear
weekdays = dte.dt.weekday
week_of_year = dte.dt.weekofyear
```

Feature Reduction

Some of the features like "instant" does not do any value addition for the analysis apart from identifying the unique record. But same thing can be induced from the row label, hence the "instant" feature can be removed.

In [ ]:

```
bikes.drop(["instant"], axis=1, inplace=True)
bikes.columns
```

Similary, "cnt" attribute can be derived from "casual" and "registered" attributes. So "cnt" is redundant information, hence can be removed.

In [ ]:

```
bikes.drop(["cnt"], axis=1, inplace=True)
bikes.columns
```

Filtering

Lets divide the data frame into two different data frames each one containing different year.

In [ ]:

```
bikes_2011 = bikes[bikes.year == 2011]
bikes_2011.head(2)
```

In [ ]:

```
bikes_2012 = bikes[bikes.year == 2012]
bikes_2012.head(2)
```

Similary several smaller data frames can be derived out of complete data set and then anlysis can be done on those data frames.

In [ ]:

```
bikes_2011_summer = bikes[(bikes.year == 2011) & (bikes.season == 2)]
bikes_2011_summer.head(2)
```

In [ ]:

```
bikes_holiday = bikes[bikes.holiday == 1]
bikes_holiday.head(2)
```

Quick stat about numeric features

Lets look at the quick statistics about this dataset.

In [ ]:

```
bikes.describe()
```

Lets look at users statistics :

In [ ]:

```
bikes.casual.describe()  #casual users stat
```

In [ ]:

```
bikes.registered.describe()  #registered users stat
```

Step3 - Group-by and Concat /Merge

Group by based on single feature

There are four seasons for which bike shairing data is collected. Lets figure out how many users are using this service as per different seasons.

In [ ]:

```
#First create the group by
df_by_seasons = bikes.groupby("season")
type(df_by_seasons)
```

In [ ]:

```
#sum("casual") group by season
df_by_seasons["casual"].sum()
```

In [ ]:

```
#sum("registered") group by season
df_by_seasons["registered"].sum()
```

In [ ]:

```
#sum("registered") group by season , sort in ascending order
df_by_seasons["registered"].sum().sort_values()
```

In [ ]:

```
#sum("registered") group by season , sort in descending order
df_by_seasons["registered"].sum().sort_values(ascending=False)
```

Combine the outcome of two group by into single data frame.

In [ ]:

```
#Create the summarized data for casual users
df_casual = df_by_seasons["casual"].sum()
df_casual
```

In [ ]:

```
#Create the summarized data for registered users
df_registered = df_by_seasons["registered"].sum()
df_registered
```

In [ ]:

```
#Combine the two frames into single one using pandas concat feature
frames = [df_casual, df_registered]
df_seasons = pd.concat(frames, axis=1)
df_seasons
```

In [ ]:

```
#Add the rowwise total for each season
df_seasons["total"] = df_seasons["casual"] + df["registered"]
df_seasons
```

Group by based on multiple features

Lets explore how more than one attributes can be used in group by clause. Average number of users for different years and months can be found as follows :

In [ ]:

```
#Create the group by based on year and months
df_by_year_months = bikes.groupby(["year", "mnth"])
```

In [ ]:

```
#Compute the average number of casual users per month per year
series_casual = df_by_year_months["casual"].mean()
series_casual
```

In [ ]:

```
#Compute the average number of registered users per month per year
series_registered = df_by_year_months["registered"].mean()
series_registered
```

In [ ]:

```
#Concatenate the results of two data frames
df_casual = pd.DataFrame(series_casual)
#df_casual
df_registered = pd.DataFrame(series_registered)
#df_registered
df_year_month = pd.merge(df_casual,df_registered, on=['year', 'mnth'])
df_year_month
```

In [ ]:

```
df_year_month["total"] = df_year_month["casual"] + df_year_month["registered"]
```

In [ ]:

```
df_year_month
```

Step 4 - Writing data frames to CSV file

Lets utilise the pandas function to write back the data frame to CSV file.

In [ ]:

```
df_seasons.head()
```

In [ ]:

```
df_seasons.to_csv("file1.csv")
```

In [ ]:

```
df_year_month.head()
```

In [ ]:

```
df_year_month.to_csv("file2.csv")
```