3-Pandas

January 14, 2021

1 Pandas

Pandas è una libreria di manipolazione e analisi dei dati che ha acquistato enorme popolarità nel tempo, pandas supporta un'enorme varietà di file che possono essere letti e modificati. La struttura cardine su cui si basa pandas è il **Dataframe** che è una collezione tabulare di dati che possono essere di tipi diversi ed identificati in colonne e righe identificative. Pandas è una libreria estremamente veloce ed efficiente dovuto al fatto che è **basata su numpy e linguaggi più performanti di python come C**.

1.1 Come installare ed usare Pandas

Pandas dovrebbe essere già presente in Anaconda, qualora però mancasse potete installarlo seguendo questo **link** come guida iniziale che fornisce molte informazioni.

Per importare pandas in genere si definisce come acronimo della libreria pd classicamente, è possibile usare qualsiasi acronimo vi venga in mente a patto che sia comprensibile!

[1]:
```
import pandas as pd
import datetime
```

1.2 Primi passi in pandas

Per capire come usare pandas incominciamo a vedere come sia possibile usarlo per effettuare delle analisi su ad esempio dati finanziari, useremo un dataset che ho scaricato da yahoo finance per vedere le sue funzionalità.

[2]:
```
#creiamo un dataframe dal file csv
nikkei = pd.read_csv("../data/Nikkei.csv", parse_dates = ['Date'])
```

Da questo momento è stato creato una variabile oggetto di tipo pandas dataframe, per avere un'idea di come essa sia composta possiamo mostrare il suo contenuto.

[3]:
```
nikkei
```

[3]:

	Date	Open	High	Low	Close	\
0	2010-01-04	10609.339844	10694.490234	10608.139648	10654.790039	
1	2010-01-05	10719.440430	10791.040039	10655.570313	10681.830078	
2	2010-01-06	10709.549805	10768.610352	10661.169922	10731.450195	
3	2010-01-07	10742.750000	10774.000000	10636.669922	10681.660156	

```
4    2010-01-08  10743.299805  10816.450195  10677.559570  10798.320313
…         …           …             …             …             …
2713 2021-01-06  27102.849609  27196.400391  27002.179688  27055.939453
2714 2021-01-07  27340.460938  27624.730469  27340.460938  27490.130859
2715 2021-01-08  27720.140625  28139.029297  27667.750000  28139.029297
2716 2021-01-12  28004.369141  28287.369141  27899.449219  28164.339844
2717 2021-01-13  28140.099609  28503.429688  28133.589844  28456.589844

         Adj Close    Volume
0     10654.790039  104400.0
1     10681.830078  166200.0
2     10731.450195  181800.0
3     10681.660156  182600.0
4     10798.320313  211800.0
…              …       …
2713  27055.939453   72700.0
2714  27490.130859   98900.0
2715  28139.029297   84900.0
2716  28164.339844   78800.0
2717  28456.589844       0.0

[2718 rows x 7 columns]
```

Per mostrare solo i primi elementi è anche possibile usare la funzione `.head()` per dare una visione più ristretta.

[4]:
```
nikkei.head()
```

[4]:
```
        Date          Open          High           Low         Close  \
0 2010-01-04  10609.339844  10694.490234  10608.139648  10654.790039
1 2010-01-05  10719.440430  10791.040039  10655.570313  10681.830078
2 2010-01-06  10709.549805  10768.610352  10661.169922  10731.450195
3 2010-01-07  10742.750000  10774.000000  10636.669922  10681.660156
4 2010-01-08  10743.299805  10816.450195  10677.559570  10798.320313

      Adj Close    Volume
0  10654.790039  104400.0
1  10681.830078  166200.0
2  10731.450195  181800.0
3  10681.660156  182600.0
4  10798.320313  211800.0
```

Per avere informazione sui tipi di dati usati per memorizzare le variabili o informazioni generali è possibile usare la funzione info.

[5]:
```
nikkei.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 2718 entries, 0 to 2717
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Date       2718 non-null   datetime64[ns]
 1   Open       2698 non-null   float64
 2   High       2698 non-null   float64
 3   Low        2698 non-null   float64
 4   Close      2698 non-null   float64
 5   Adj Close  2698 non-null   float64
 6   Volume     2698 non-null   float64
dtypes: datetime64[ns](1), float64(6)
memory usage: 148.8 KB
```

Come è possibile notare pandas ci dice già **quanti sono i valori nulli o non null** e specifica il formato in cui sono salvati i dati, notate bene che **i formati usati sono uguali a quelli in numpy**, ciò non deve stupire poiché è basata su di esso la libreria, inoltre ci fornisce informazioni sulla **memoria occupata** per salvare il dataframe.

```
I valori nulli sono definiti come tutti quei valori che non sono stati definiti oppure che non
```

1.3 Operazioni sul DataFrame

Sul dataframe è possibile applicare numerose operazioni tra cui selezionare specifici elementi, calcolare le medie, selezionare elementi e fare molto altro.

[6]:
```
#select only the column Open
nikkei["Open"].head()
```

[6]:
```
0    10609.339844
1    10719.440430
2    10709.549805
3    10742.750000
4    10743.299805
Name: Open, dtype: float64
```

[7]:
```
#Select only max e min value of AdjClose and calculate mean
print("Prezzo Massimo Adj close", nikkei["Adj Close"].max(), "USD")
print("Prezzo Minimo Adj close", nikkei["Adj Close"].min(), "USD")
print("Media di tutti i valori Adj Close", nikkei["Adj Close"].mean(), "USD")
#possibile selezionare anche la riga contenente il valore
print("Tutti i valori il giorno in cui Adj Close era maggiore o uguale al min␣
 ↪storico:\n")
display(nikkei[nikkei["Adj Close"] >= nikkei["Adj Close"].min()])
```

```
Prezzo Massimo Adj close 28456.589844 USD
Prezzo Minimo Adj close 8160.009766 USD
Media di tutti i valori Adj Close 16437.871096161947 USD
Tutti i valori il giorno in cui Adj Close era maggiore o uguale al min storico:
```

```
           Date          Open          High           Low         Close  \
0    2010-01-04  10609.339844  10694.490234  10608.139648  10654.790039
1    2010-01-05  10719.440430  10791.040039  10655.570313  10681.830078
2    2010-01-06  10709.549805  10768.610352  10661.169922  10731.450195
3    2010-01-07  10742.750000  10774.000000  10636.669922  10681.660156
4    2010-01-08  10743.299805  10816.450195  10677.559570  10798.320313
…           …             …             …             …             …
2713 2021-01-06  27102.849609  27196.400391  27002.179688  27055.939453
2714 2021-01-07  27340.460938  27624.730469  27340.460938  27490.130859
2715 2021-01-08  27720.140625  28139.029297  27667.750000  28139.029297
2716 2021-01-12  28004.369141  28287.369141  27899.449219  28164.339844
2717 2021-01-13  28140.099609  28503.429688  28133.589844  28456.589844

         Adj Close    Volume
0     10654.790039  104400.0
1     10681.830078  166200.0
2     10731.450195  181800.0
3     10681.660156  182600.0
4     10798.320313  211800.0
…              …       …
2713  27055.939453   72700.0
2714  27490.130859   98900.0
2715  28139.029297   84900.0
2716  28164.339844   78800.0
2717  28456.589844       0.0

[2698 rows x 7 columns]
```

[8]:
```
#possibile selezionare anche gli elementi come in numpy usando loc o iloc
#prime mille righe e 6 colonne
nikkei.iloc[0:1000, 1:6]
```

[8]:
```
             Open          High           Low         Close     Adj Close
0    10609.339844  10694.490234  10608.139648  10654.790039  10654.790039
1    10719.440430  10791.040039  10655.570313  10681.830078  10681.830078
2    10709.549805  10768.610352  10661.169922  10731.450195  10731.450195
3    10742.750000  10774.000000  10636.669922  10681.660156  10681.660156
4    10743.299805  10816.450195  10677.559570  10798.320313  10798.320313
..            …             …             …             …             …
995  15091.450195  15109.679688  14933.549805  15005.730469  15005.730469
996  15038.639648  15088.120117  14952.830078  14980.160156  14980.160156
997  15164.339844  15383.910156  15159.919922  15383.910156  15383.910156
998  15112.700195  15112.700195  14853.830078  15007.059570  15007.059570
999  15132.230469  15143.879883  14764.570313  14914.530273  14914.530273

[1000 rows x 5 columns]
```

[9]:
```
#altro modo più intuitivo
nikkei[["Open", "High", "Low", "Close", "Adj Close"]]
```

[9]:

	Open	High	Low	Close	Adj Close
0	10609.339844	10694.490234	10608.139648	10654.790039	10654.790039
1	10719.440430	10791.040039	10655.570313	10681.830078	10681.830078
2	10709.549805	10768.610352	10661.169922	10731.450195	10731.450195
3	10742.750000	10774.000000	10636.669922	10681.660156	10681.660156
4	10743.299805	10816.450195	10677.559570	10798.320313	10798.320313
…	…	…	…	…	…
2713	27102.849609	27196.400391	27002.179688	27055.939453	27055.939453
2714	27340.460938	27624.730469	27340.460938	27490.130859	27490.130859
2715	27720.140625	28139.029297	27667.750000	28139.029297	28139.029297
2716	28004.369141	28287.369141	27899.449219	28164.339844	28164.339844
2717	28140.099609	28503.429688	28133.589844	28456.589844	28456.589844

[2718 rows x 5 columns]

Sono presenti numerose altre operazioni che sono possibili da fare con pandas, alcuni librerie di backtesting per trading o investimento sono basate su pandas per fare un esempio, qualora aveste dubbi o volete sapere quali altri funzionalità abbia fate un salto sulla **documentazione**.

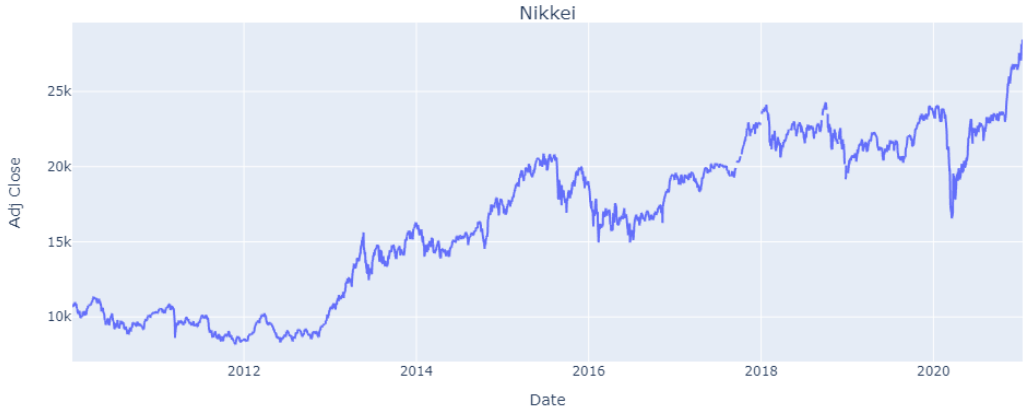
1.4 Plot pandas with plotly

Per rappresentare il dataframe in genere è possibile usare la libreria **matplotlib**, ma poiché si tratta di dati finanziari preferisco mostrare la libreria già accennata **plotly**, vediamo alcuni esempi.

Per poter usare plotly su jupyter lab è necessario usare i seguenti comandi in anaconda:
 conda install nodejs
 jupyter labextension install jupyterlab-plotly@4.14.3
Per problemi consultate la guida .

[10]:
```
import plotly.express as px
import plotly.graph_objects as go

#classico grafico di una singola linea
fig = px.line(nikkei, x = "Date", y="Adj Close")
#create title and center it
fig.update_layout(
    title={
        'text': "Nikkei",
        'y':0.9,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top'})
#show figure
fig.show()
```



Plotly però in particolare permette di poter definire altri tipi di plot in particolare in ambito finanziario come i candlesticks.

[11]:
```
#create candlestick chart
fig = go.Figure(data=[go.Candlestick(x = nikkei["Date"],
                open=nikkei['Open'],
                high=nikkei['High'],
                low=nikkei['Low'],
                close=nikkei['Close'])])
#remove the rangeslider and add title centered
fig.update_layout(xaxis_rangeslider_visible=False,
                  title={'text': "Nikkei",
                         'y':0.9,
                         'x':0.5,
                         'xanchor': 'center',
                         'yanchor': 'top'})

fig.show()
```

Questa lezione di pandas vuole darvi sono le basi per sull'utilizzo di questa libreria, qualora voi abbiate dubbi o chiarimenti potete consultare questa recente **guida** su come iniziare e come sempre consultare la **guida completa**. *** COMPLIMENTI AVETE COMPLETATO LA LEZIONE SULLA LIBRERIA PANDAS!