Python adatábrázolás

Az adatok

Három a mesterséges intelligenciához kapcsolható kifejezés a "data science, machine learning és deep learning" népszerűségi adatait (Egy tudományos keresőmotor keresési számai).
Python csomagok: Pandas

Pandas

A pandas egy Python csomag, amely gyors, rugalmas és kifejező adatstruktúrákat kínál, amelyek célja a „relációs” vagy „címkézett” adatokkal való egyszerű és intuitív munka. Főleg táblázatos adatok kezelésére szolgál pl sql tábla. Kezelhető idősoros adat vagy cimkézett mátrix is vele.
két fő adastruktúrája series (1D) dataFrame(2D) NumPy szükséges

1 feladat importáljuk és vizsgáljuk meg az adatokat

```
import pandas as pd
df = pd.read_csv('temporal.csv')

df.head(10)
```

	date	data science	machine learning	deep learning	categorical
0	2004-01-01	12	18	4	1
1	2004-02-01	12	21	2	1
2	2004-03-01	9	21	2	1
3	2004-04-01	10	16	4	1
4	2004-05-01	7	14	3	1
5	2004-06-01	9	17	3	1

6	2004-07-01	9	16	3	1
7	2004-08-01	7	14	3	1
8	2004-09-01	10	17	4	1
9	2004-10-01	8	17	4	1

```
df.tail(10)
```

	date	data science	machine learning	deep learning	categorical
184	2019-05-01	76	92	97	0
185	2019-06-01	80	93	96	0
186	2019-07-01	85	92	96	0
187	2019-08-01	88	93	92	0
188	2019-09-01	95	100	100	0
189	2019-10-01	90	98	98	0
190	2019-11-01	87	97	96	0
191	2019-12-01	81	89	91	0
192	2020-01-01	94	94	93	1
193	2020-02-01	100	99	99	1

```
print(df)
```

```
           date  data science  machine learning  deep learning  categorical
0    2004-01-01            12                18
```

```
4              1
1    2004-02-01             12                   21
2              1
2    2004-03-01              9                   21
2              1
3    2004-04-01             10                   16
4              1
4    2004-05-01              7                   14
3              1
..           ...            ...                  ...
...            ...
189  2019-10-01             90                   98
98             0
190  2019-11-01             87                   97
96             0
191  2019-12-01             81                   89
91             0
192  2020-01-01             94                   94
93             1
193  2020-02-01            100                   99
99             1

[194 rows x 5 columns]
```

Első lehetséges vizualizáció tehát a 2D táblázatos megjelenítése az adatoknak dátum + 4 változó: 3 AI fogalom + 1 kategorikus vált (azaz 0 vagy 1, hogy kategorikus változók használatát/lehetőségeit is bemutathassuk)
Vizsgáljuk meg a változók leíró statisztikáit!

```
df.describe() #pandas dataframe descStat függvénye!
```

	data science	machine learning	deep learning	categorical
count	194.000000	194.000000	194.000000	194.000000
mean	20.953608	27.396907	24.231959	0.257732
std	23.951006	28.091490	34.476887	0.438517
min	4.000000	7.000000	1.000000	0.000000
25%	6.000000	9.000000	2.000000	0.000000
50%	8.000000	13.000000	3.000000	0.000000

75%	26.750000	31.500000	34.000000	1.000000
max	100.000000	100.000000	100.000000	1.000000

Átlag, szórás, min, max stb... Figyeljük meg, hogy a kategorikus változónknak is megadhatóa e statisztikái, kérdés, hogy van-e értelme... Vizsgáljuk meg, hogy az egyes oszlopok milyen típusú adatokat tartalmaznak.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 194 entries, 0 to 193
Data columns (total 5 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   date              194 non-null    object
 1   data science      194 non-null    int64 
 2   machine learning  194 non-null    int64 
 3   deep learning     194 non-null    int64 
 4   categorical       194 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 7.7+ KB
```

5 oszlop: 1 obj + 4 int64 194 megfigyelés
Úgy tűnik, hogy a dátum nem szám adatípusú -> stringként viselkedik.

Következő táblázatos vizualizációs lehetőség!

Gyakran felmerül az igény, hogy az adatainkat speciális módon szeretnénk megjeleníten a táblázatban (pl részletesebb pontosabb info érdekében), pl szeretnénk látni az adatok mellett a mértékegységét/ pénznemét/százalékot. Mindezt úgyhogy az adat maradjon számként tárolva ne váljon stringgé!
Nézzünk egy példát!

- Adjunk speciális megjelenést a dátumnak pl csak a hómap és év látszódjon
- data science vált legyen forintban
- machine learning két tizedere megjelenített %

Ez csak példa nincs sok értelme. DE az adat nem változik meg a parancsoksorán, azaz ez csak a megjelnítésükre van hatással, akár egy maszk.

```
format_dict = {'data science':'{0:,.2f} HUF',
'date':'{:%m-%Y}', 'machine learning':'{:.2%}'}
#We make sure that the Month column has datetime
```

```
format
df['date'] = pd.to_datetime(df['date'])
#We apply the style to the visualization
df.head().style.format(format_dict) # a létrehozott maszkot alkalmazzuk a megjelenítés során
```

	date	data science	machine learning	deep learning	categorical
0	01-2004	12.00 HUF	1800.00%	4	1
1	02-2004	12.00 HUF	2100.00%	2	1
2	03-2004	9.00 HUF	2100.00%	2	1
3	04-2004	10.00 HUF	1600.00%	4	1
4	05-2004	7.00 HUF	1400.00%	3	1

Következő táblázatos vizualizációs lehetőség!¶

Emeljük ki a táblázatban a Min és Max értékek értékeket! Ez is csak egy maszk és a megjelenített táblára vonatkozik (oszloponként)

```
format_dict = {'date':'{:%m-%Y}'} # A maszk deklarását írjuk át csak a dátum oszlopra, így több értelme lesz a használatának.
df.head(10).style.format(format_dict).highlight_max(color='darkgreen').highlight_min(color='#ff0000')
```

	date	data science	machine learning	deep learning	categorical
0	01-2004	12	18	4	1
1	02-2004	12	21	2	1
2	03-2004	9	21	2	1
3	04-2004	10	16	4	1
4	05-2004	7	14	3	1
5	06-2004	9	17	3	1
6	07-2004	9	16	3	1
7	08-2004	7	14	3	1

8	09-2004	10	17	4	1
9	10-2004	8	17	4	1

Használhatunk színátmeneteket is.

```
df.head(10).style.format(format_dict).background_gradient(
    subset=['data science', 'machine learning'], cmap='BuGn')
```

	date	data science	machine learning	deep learning	categorical
0	01-2004	12	18	4	1
1	02-2004	12	21	2	1
2	03-2004	9	21	2	1
3	04-2004	10	16	4	1
4	05-2004	7	14	3	1
5	06-2004	9	17	3	1
6	07-2004	9	16	3	1
7	08-2004	7	14	3	1
8	09-2004	10	17	4	1
9	10-2004	8	17	4	1

Vagy sávos színezést is

```
df.head(10).style.format(format_dict).bar(color='red', subset=['data science', 'deep learning'])
```

	date	data science	machine learning	deep learning	categorical
0	01-2004	12	18	4	1
1	02-2004	12	21	2	1

2	03-2004	9	21	2	1
3	04-2004	10	16	4	1
4	05-2004	7	14	3	1
5	06-2004	9	17	3	1
6	07-2004	9	16	3	1
7	08-2004	7	14	3	1
8	09-2004	10	17	4	1
9	10-2004	8	17	4	1

Vagy kombinálhatjuk ezeket pl. színgradiens és max kiemelés

```
df.head(10).style.format(format_dict).background_gradient(
    subset=['data science', 'machine learning'], cmap='BuGn').highlight_max(color='yellow')
```

	date	data science	machine learning	deep learning	categorical
0	01-2004	12	18	4	1
1	02-2004	12	21	2	1
2	03-2004	9	21	2	1
3	04-2004	10	16	4	1
4	05-2004	7	14	3	1
5	06-2004	9	17	3	1
6	07-2004	9	16	3	1
7	08-2004	7	14	3	1

8	09-2004	10	17	4	1
9	10-2004	8	17	4	1

Learn more about styling visualizations with Pandas here: https://pandas.pydata.org/pandas-docs/stable/user_guide/style.html

Matplotlib

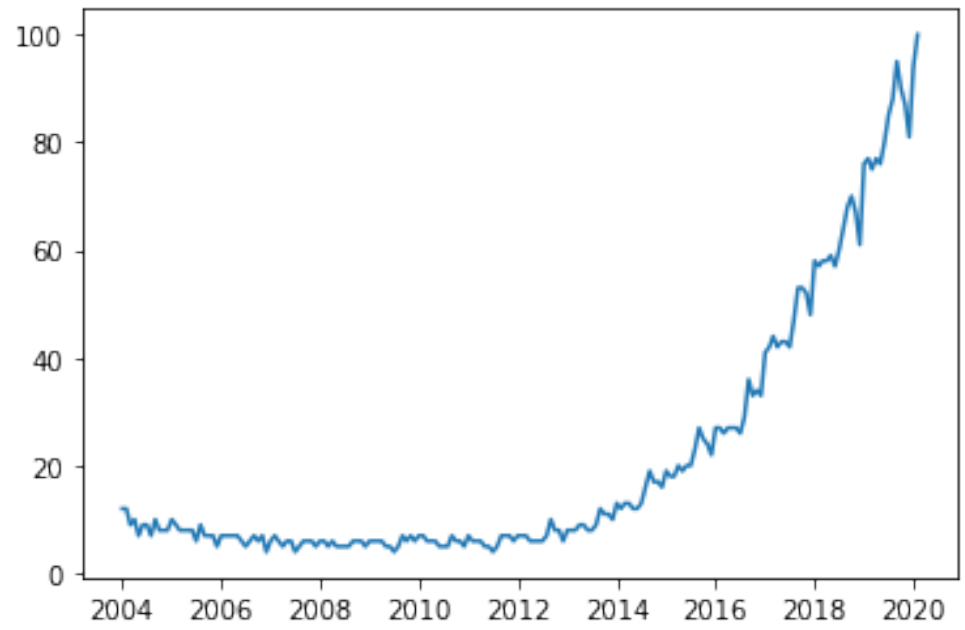
Az adatok táblázatos megjelenítése után nézzük a grafikus megjelnítést. A Matplotlib a legalapvetőbb könyvtár az adatok grafikus megjelenítésére. Sok olyan grafikont tartalmaz, amelyekre gondolhatunk. A Matplotlib diagramjai két fő összetevőből állnak, az ábrából/figure (ahol a tengelyeket, címeket és dolgokat rajzoljuk ki) és a tengelyekből (a diagram területét határoló vonalakból) és.

vonaldiagram

Most hozzuk létre a lehető legegyszerűbb grafikont:

```
import matplotlib.pyplot as plt
plt.plot(df['date'], df['data science'], label='data science')
```

```
[<matplotlib.lines.Line2D at 0x2b93c9a26c8>]
```

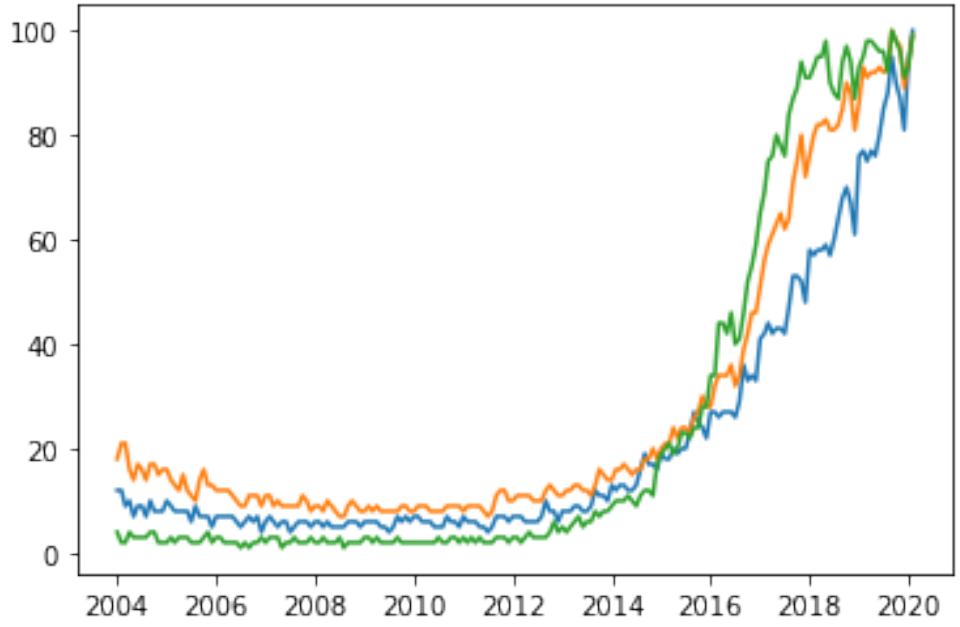


A paramétercímke a jelmagyarázatot jelzi. Ez nem jelenti azt, hogy meg is jelenik, ehez más parancsot kell használnunk, amelyet később elmagyarázok.
Ugyanazon gráfban több változó grafikonját is elkészíthetjük, és így

összehasonlíthatjuk őket.

```
plt.plot(df['date'], df['data science'], label='data
science')
plt.plot(df['date'], df['machine learning'],
label='machine learning')
plt.plot(df['date'], df['deep learning'], label='deep
learning')
```

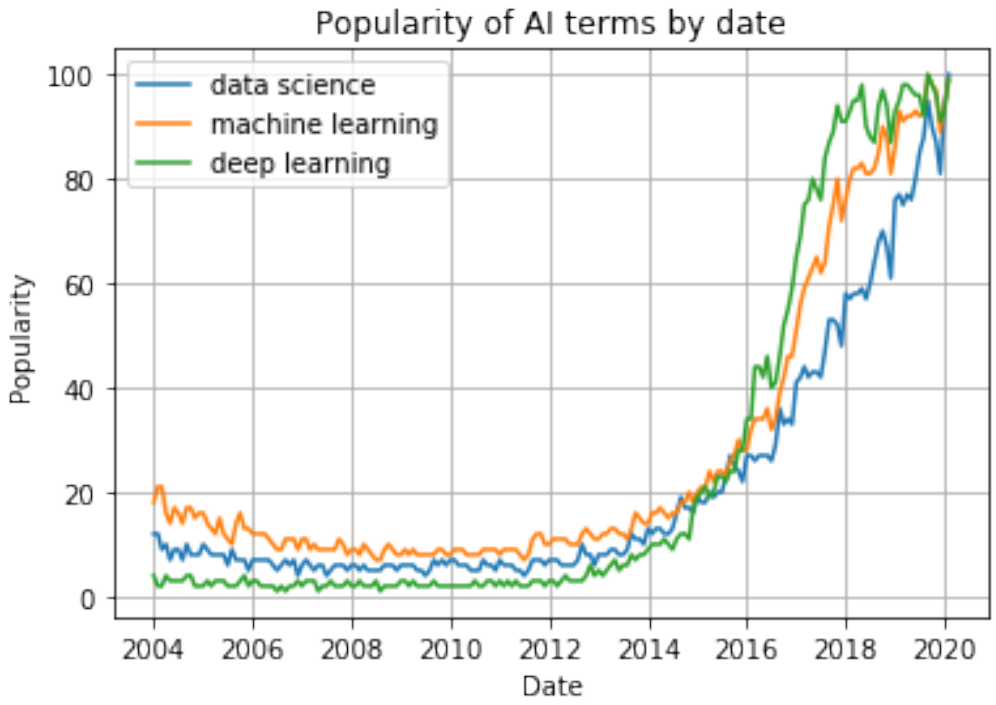
```
[<matplotlib.lines.Line2D at 0x2b93d14a3c8>]
```



OK DE nem világos, hogy az egyes színek melyik változót képviselik. Adjunk hozzá jelmagyarázatot is.

```
plt.plot(df['date'], df['data science'], label='data
science')
plt.plot(df['date'], df['machine learning'],
label='machine learning')
plt.plot(df['date'], df['deep learning'], label='deep
learning')
plt.xlabel('Date')
plt.ylabel('Popularity')
plt.title('Popularity of AI terms by date')
plt.grid(True)
plt.legend()
```

```
<matplotlib.legend.Legend at 0x2b93d1db188>
```

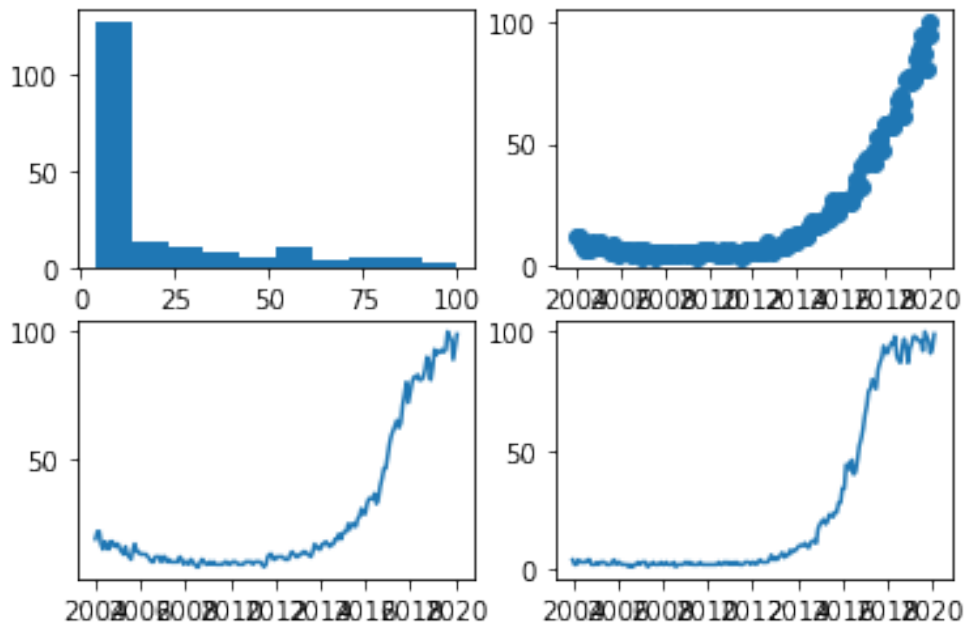


többábra egyszerre

Több ábrát is készíthetünk egy ábrán. Ez nagyon jól hasznos a diagramok összehasonlításához vagy a többféle diagramból származó adatok egyszerű megosztásához egyetlen képpel. pl hisztogram és adatsor... v több adatsor összehasonlítására...

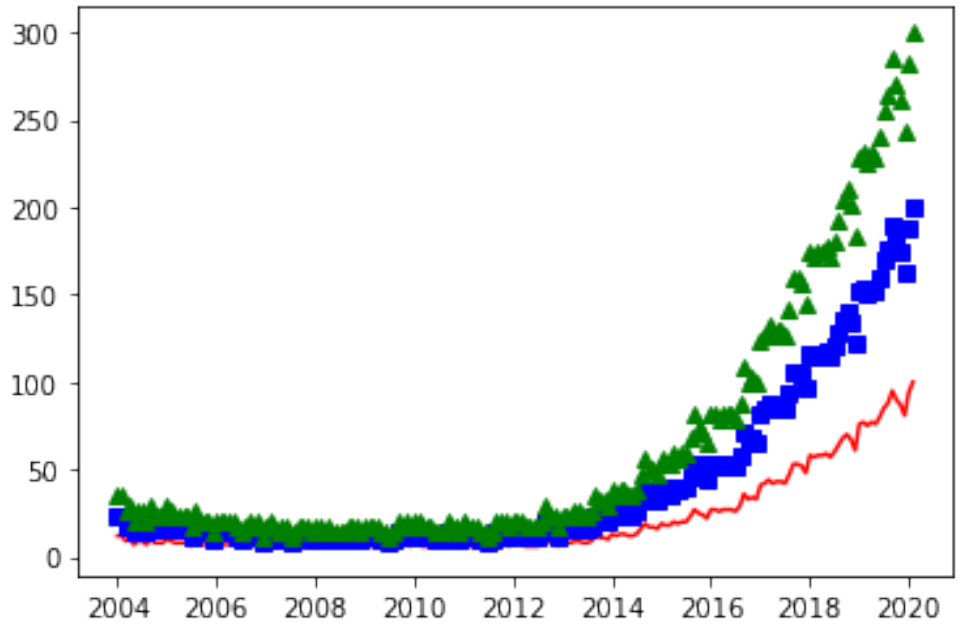
```
fig, axes = plt.subplots(2,2)
axes[0, 0].hist(df['data science'])
axes[0, 1].scatter(df['date'], df['data science'])
axes[1, 0].plot(df['date'], df['machine learning'])
axes[1, 1].plot(df['date'], df['deep learning'])
```

```
[<matplotlib.lines.Line2D at 0x2b93d30e408>]
```



```
plt.plot(df['date'], df['data science'], 'r-')
plt.plot(df['date'], df['data science']*2, 'bs')
plt.plot(df['date'], df['data science']*3, 'g^')
```

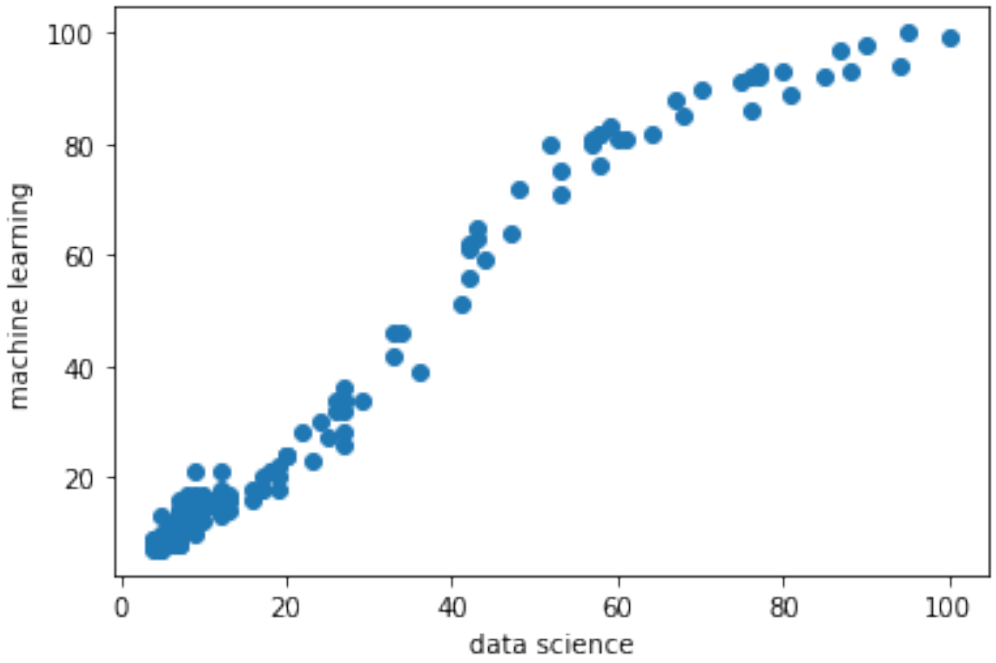
```
[<matplotlib.lines.Line2D at 0x2b93d43adc8>]
```



x-y scatter plot

```
plt.scatter(df['data science'], df['machine
learning'])
plt.xlabel('data science')
plt.ylabel('machine learning')
```

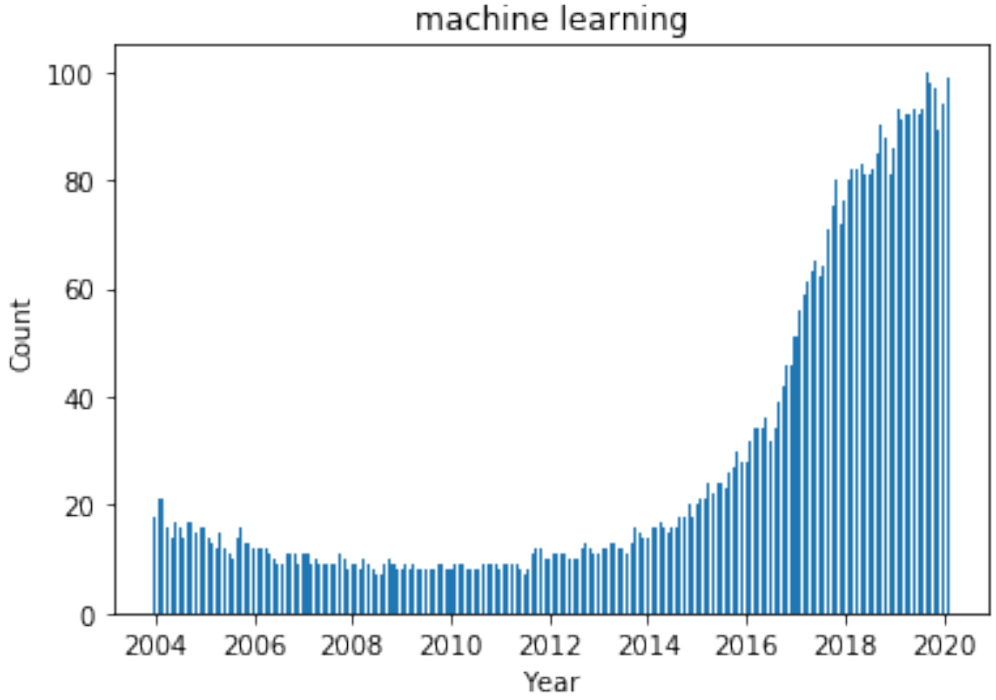
```
Text(0, 0.5, 'machine learning')
```



Barchart

```
plt.bar(df['date'], df['machine learning'], width=20)
plt.xlabel('Year')
plt.ylabel('Count')
plt.title('machine learning')
```

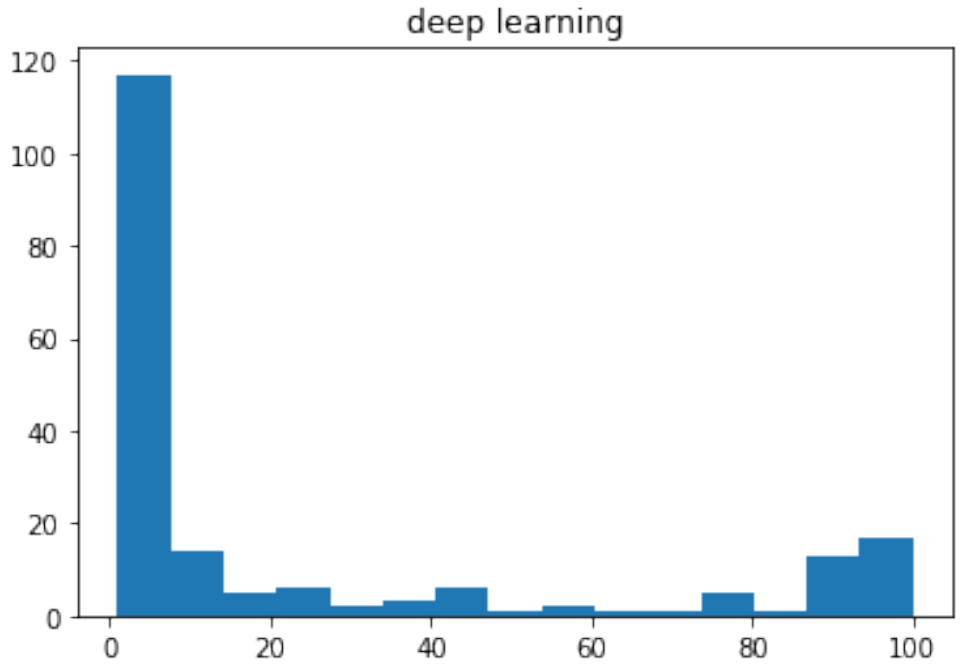
```
Text(0.5, 1.0, 'machine learning')
```



hisztogram

```
plt.hist(df['deep learning'], bins=15)
plt.title('deep learning')
```

```
Text(0.5, 1.0, 'deep learning')
```



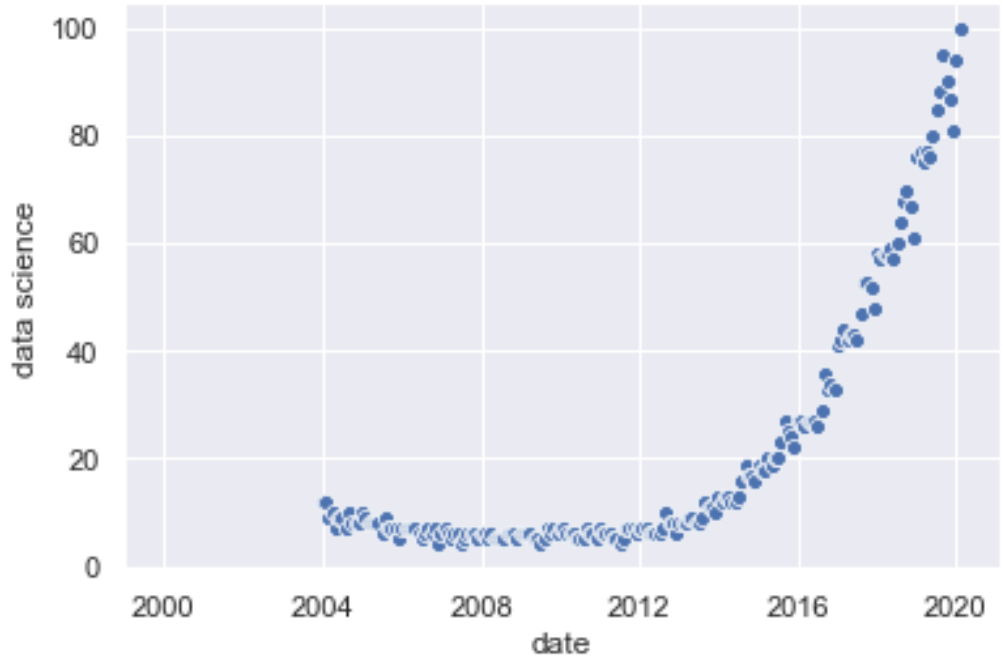
t

Seaborn

A Seaborn a Matplotlib alapú könyvtár. Alapvetően az ad nekünk szebb grafikát és funkciókat, hogy összetett típusú grafikákat készítsünk egyetlen kódsorral. Importáljuk a könyvtárat, és inicializáljuk a grafika stílusát az sns.set () paranccsal, e parancs nélkül a grafika továbbra is ugyanolyan stílusú lenne, mint a Matplotlib.
Nem elérhető az online verzióban.

```
import seaborn as sns
sns.set()
sns.scatterplot(df['date'], df['data science'])
```

```
<matplotlib.axes._subplots.AxesSubplot at
0x2b93fcc4788>
```

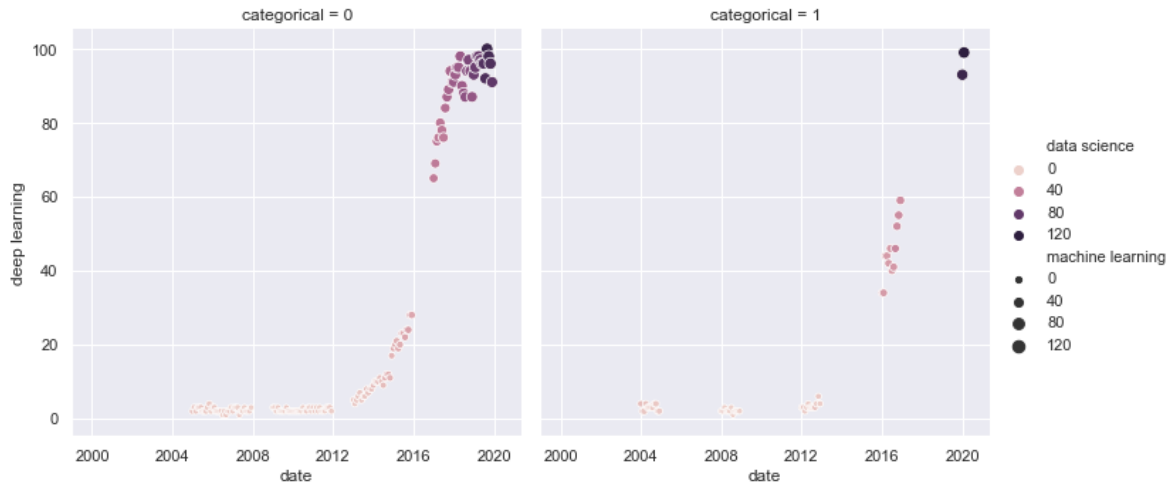


kategoriák felhasználása

Ugyanazon grafikonon több mint két változó adatait is hozzáadhatjuk. Ehhez színeket és méreteket használunk. A kategória oszlop értéke szerint más grafikont is készítünk:

```
sns.relplot(x='date', y='deep learning', hue='data science', size='machine learning', col='categorical', data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x2b93fd30f48>
```



Heatmap/korrel

A Seaborn egyik legnépszerűbb grafikája a hőtérkép. Nagyon gyakori, hogy az adathalmaz változói közötti összes összefüggés megjelenítésére használják Tulajdonképpen egy korrelációs grafikon

```
sns.heatmap(df.corr(), annot=True, fmt='.2f')
```

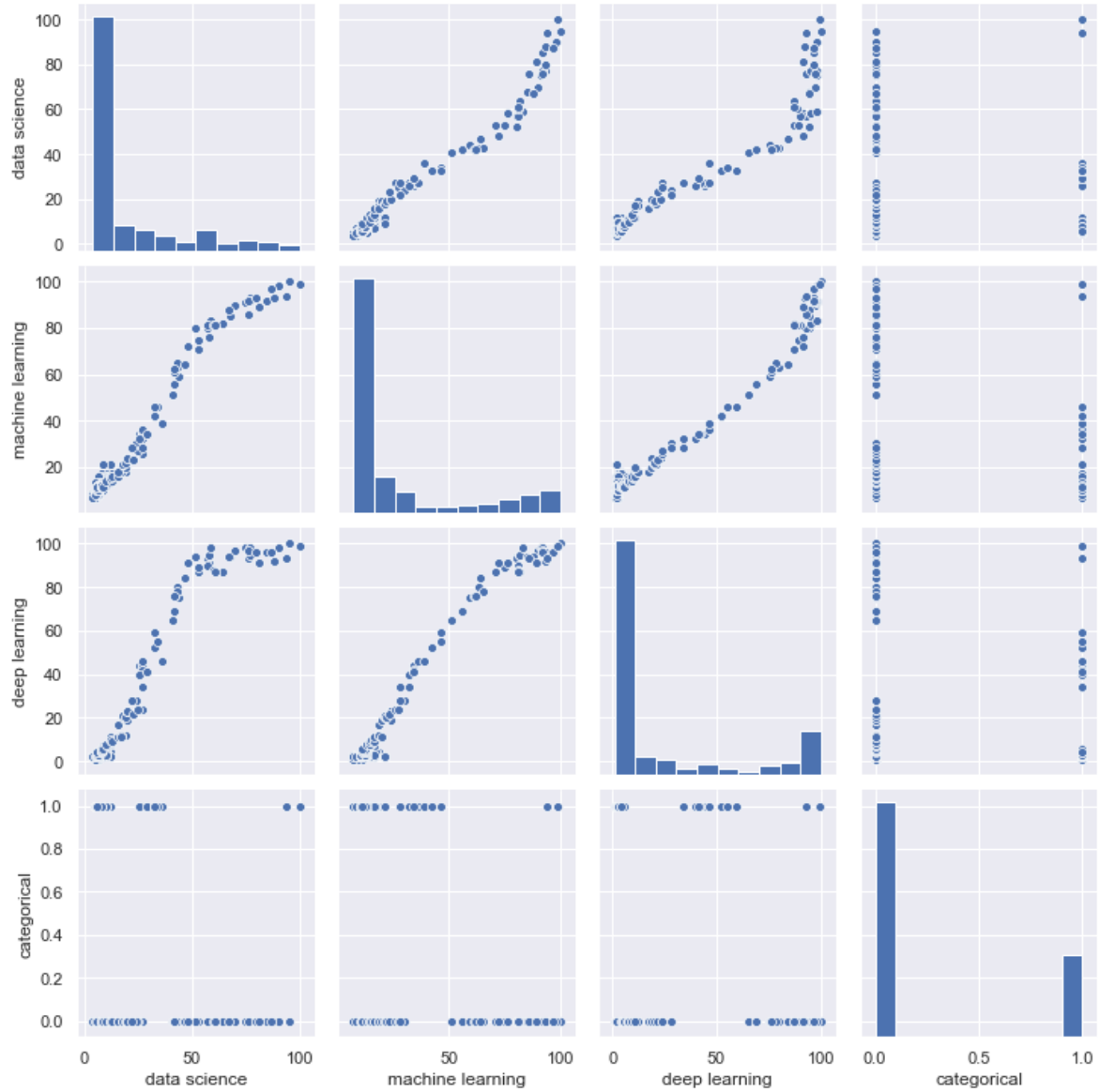
```
<matplotlib.axes._subplots.AxesSubplot at
0x2b93fe33748>
```

pairplot

A másik legnépszerűbb a pairplot, amely megmutatja nekünk az összes változó közötti kapcsolatokat. Az adatok dimenziójának növelésével a feldolgozási idő exponenciálisan nő!

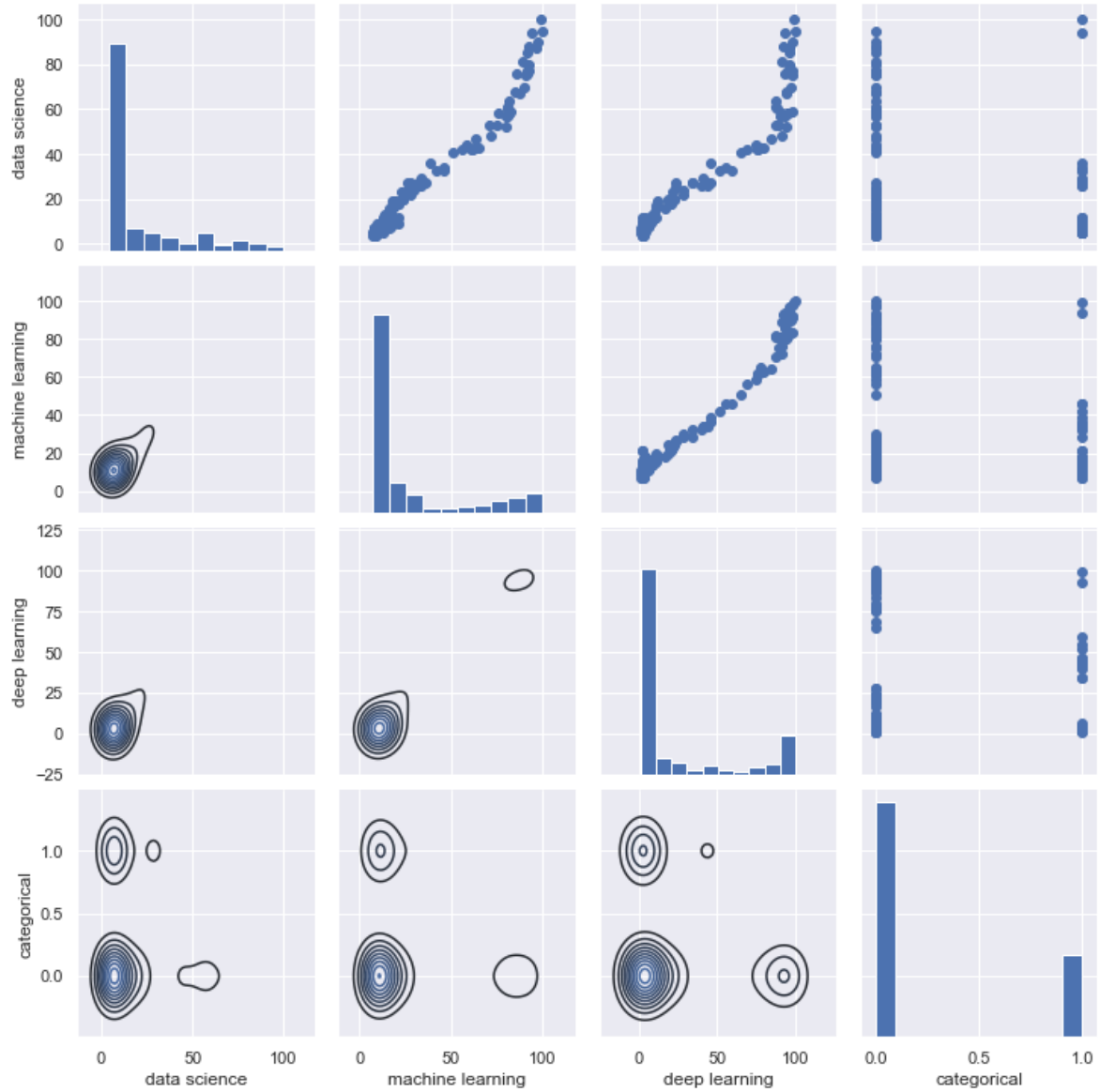
```
sns.pairplot(df)
```

```
<seaborn.axisgrid.PairGrid at 0x2b93d1033c8>
```




```
g = sns.PairGrid(df)
g.map_diag(plt.hist)
g.map_upper(plt.scatter)
g.map_lower(sns.kdeplot) # kernel density estimate, diagram egy módszer a megfigyelések eloszlásának megjelenítésére egy adathalmazban, egy hisztogramhoz hasonlóan. A KDE egy valószínűségi sűrűségi görbe segítségével ábrázolja az adatokat egy vagy több dimenzióban.
```

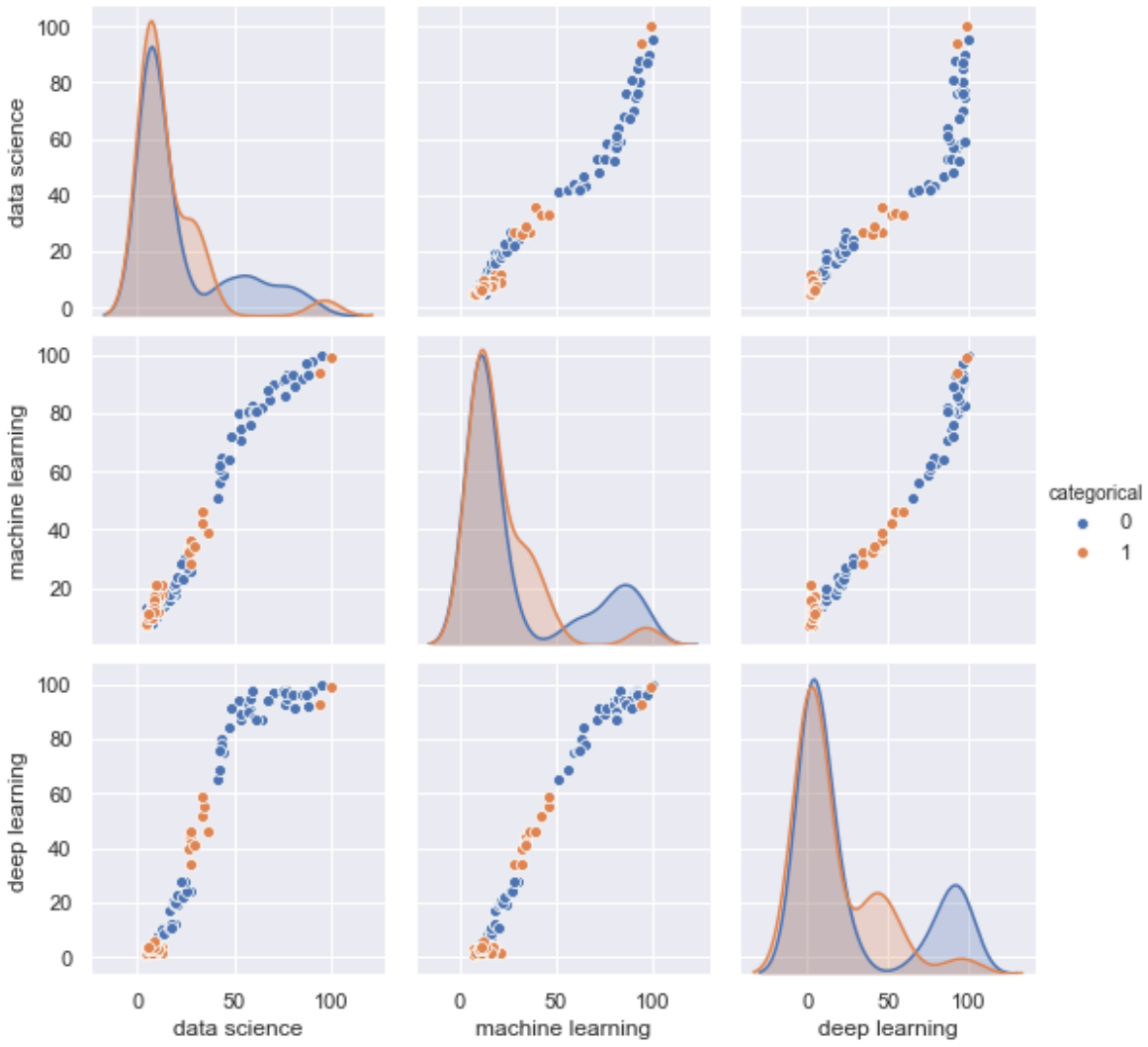
```
<seaborn.axisgrid.PairGrid at 0x2b941837c88>
```



pairplot és ketg

```
sns.pairplot(df, hue='categorical')
```

```
<seaborn.axisgrid.PairGrid at 0x2b942260108>
```

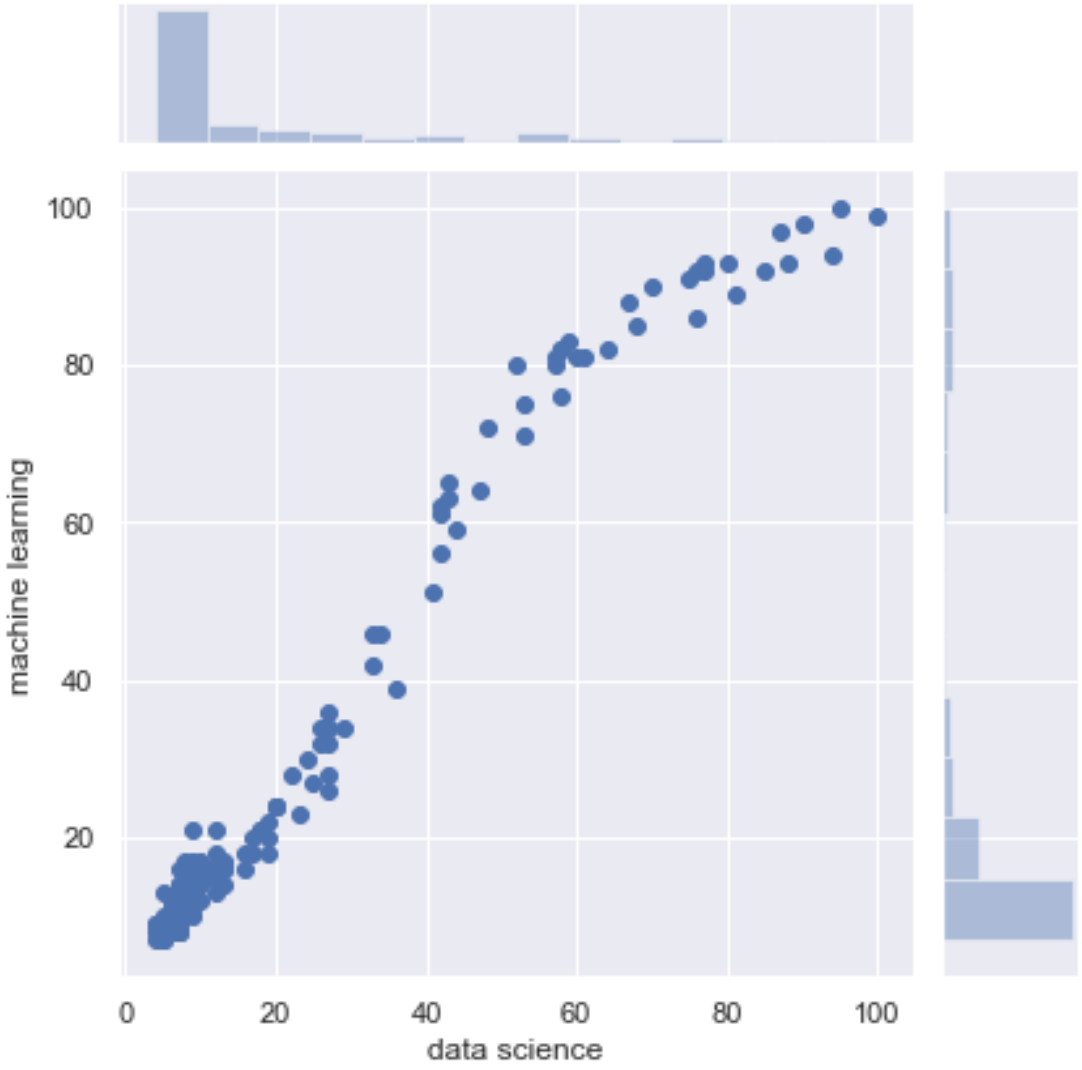


jointplot

lehetővé teszi számunkra, hogy a x-y scatterplot vagy szórásképet a két változó hisztogramjával együtt lássuk

```
sns.jointplot(x='data science', y='machine learning', data=df)
```

```
<seaborn.axisgrid.JointGrid at 0x2b942b39548>
```

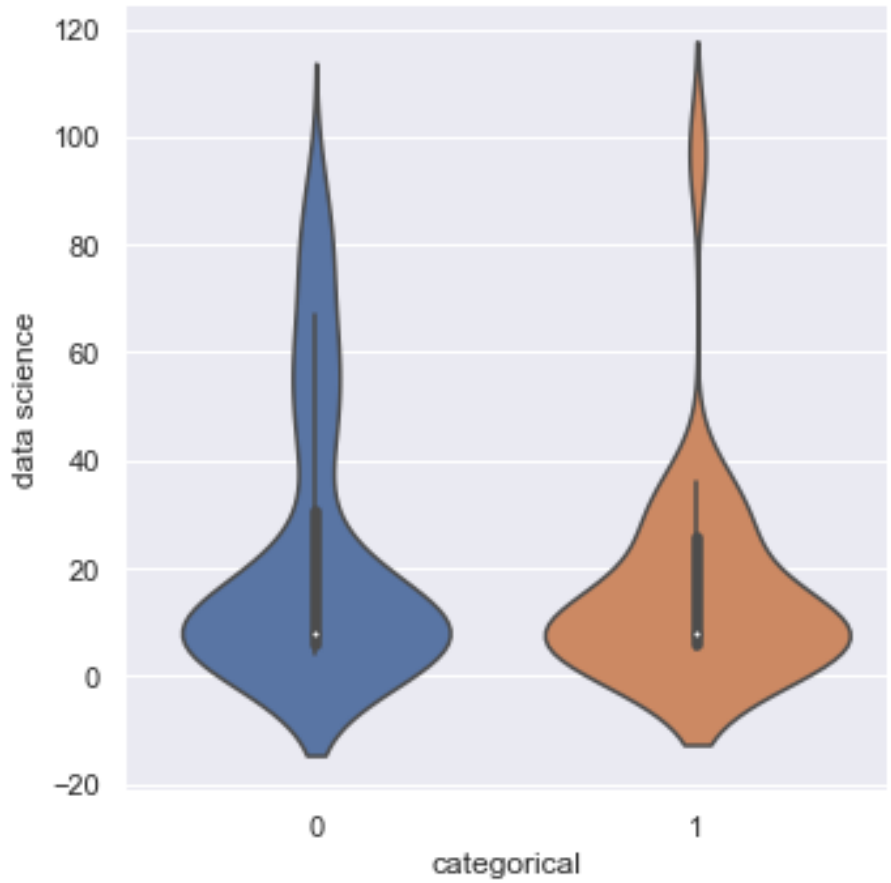


ViolinPlot

A ViolinPlot informatívabb, mint a sima box plot. Míg a dobozdiagram csak összefoglaló statisztikákat mutat, például átlag/medián és interkvartilis tartományokat, a hegedűdiagram az adatok teljes eloszlását mutatja. A különbség különösen akkor hasznos, ha az adateloszlás multimodális (több mint egy csúcs). Ebben az esetben egy hegedűdiagram különböző csúcsok jelenlétét, helyzetét és relatív amplitúdóját mutatja.

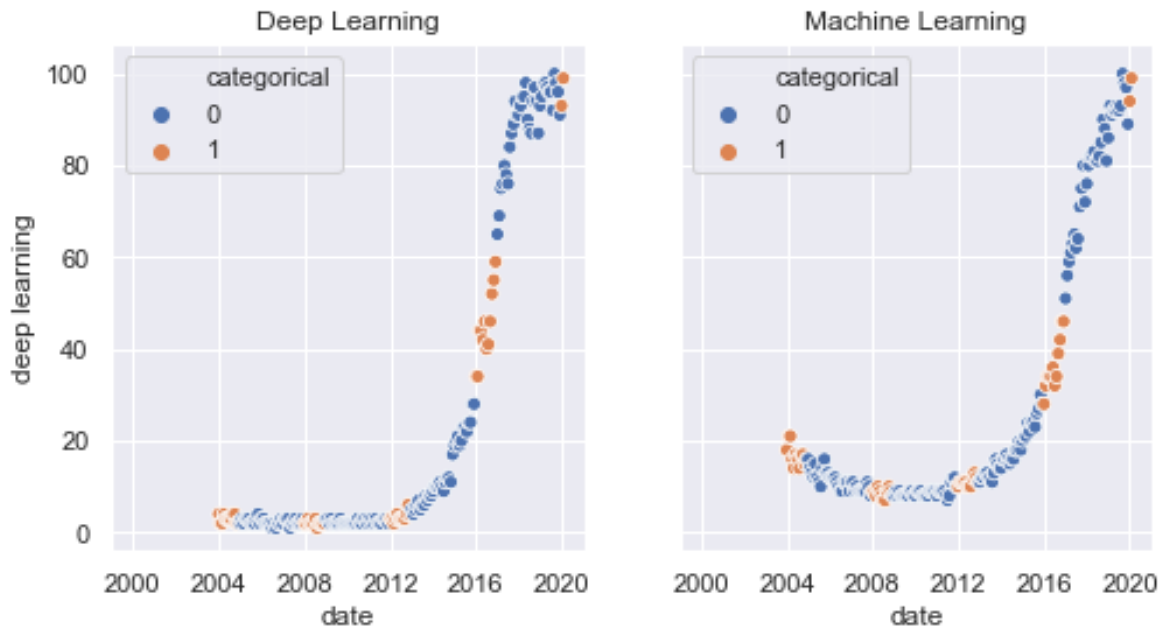
```
sns.catplot(x='categorical', y='data science',
kind='violin', data=df)
```

```
<seaborn.axisgrid.FacetGrid at 0x2b942a885c8>
```



Több képet is létrehozhatunk egy képen, mint a Matplotlib esetében

```
fig, axes = plt.subplots(1, 2, sharey=True,
figsize=(8, 4))
sns.scatterplot(x="date", y="deep learning",
hue="categorical", data=df, ax=axes[0])
axes[0].set_title('Deep Learning')
sns.scatterplot(x="date", y="machine learning",
hue="categorical", data=df, ax=axes[1])
axes[1].set_title('Machine Learning')
Text(0.5, 1.0, 'Machine Learning')
```



https://seaborn.pydata.org/examples/index.html

Bokeh

A Bokeh egy könyvtár, amely lehetővé teszi interaktív grafikák létrehozását. Exportálhatjuk őket egy HTML dokumentumba, amelyet megoszthatunk bárkivel, akinek van webböngészője.
Ez egy nagyon hasznos könyvtár, fontos a grafikában való keresés, vagy szeretnénk nagyítani és mozogni a grafikán. Vagy amikor meg akarjuk osztani ezeket.
Importáljuk a könyvtárat, és meghatározzuk azt a fájlt, amelybe menteni szeretnénk a grafikont

```
from bokeh.plotting import figure, output_file, save
output_file('data_science_popularity.html')
p = figure(title='data science', x_axis_label='date',
y_axis_label='data science')
p.line(df['date'], df['data science'],
legend_label='popularity', line_width=2)
save(p)
```

Adding multiple graphics to a single file

```
from bokeh.layouts import gridplot
output_file('multiple_graphs.html')
```

```
s1 = figure(width=250, plot_height=250, title='data
science')
s1.circle(df['date'], df['data science'], size=10,
color='navy', alpha=0.5)
s2 = figure(width=250, height=250,
x_range=s1.x_range, y_range=s1.y_range,
title='machine learning') #share both axis range
s2.triangle(df['date'], df['machine learning'],
size=10, color='red', alpha=0.5)
s3 = figure(width=250, height=250,
x_range=s1.x_range, title='deep learning') #share
only one axis range
s3.square(df['date'], df['deep learning'], size=5,
color='green', alpha=0.5)
p = gridplot([[s1, s2, s3]])
save(p)
```

https://docs.bokeh.org/en/latest/docs/gallery.html