

analytics and visualization sense

Descriptive analytics: analyse and derive insights from past data.

Predictive analytics: study trends and predict what will happen in the future.

Prescriptive analytics: analyzed the data, predict what might happen and provides insights into what steps should be taken based on the available data and what the impact of these decisions would be

right data: 百分比数据，各部分数字总和得等与100%

right chart: 使用正确的图表刻画数据，比如饼状图->柱状图

right label: 如果不使用label就无法展示信息

right axis: 一个维度能反映的区别不要用两个维度来做

不要用3D图装逼

一些小设置

%matplotlib inline。内嵌绘图，并且可以省略掉plt.show()这一步

import matplotlib.pyplot as plt

使用既定画图风格: matplotlib.style.available; plt.style.use("")

自定义喜欢的风格: matplotlib.rcParams; matplotlib.rcParams["key"]=xxx来设置参数

画图函数

plt.figure(figsize=(8,6))。设置一个canvas画板

fig=plt.figure()。创建画布

plt.annotate("text", xy, xytext, arrowprops); xy=(x,y)，坐标根据data的值设置，xytext, arrowprops设置箭头，如{facecolor="black",shrink=0.05}

df.plot(kind="TypeOfPlot", x="x axis name", y="y axis name", color="color for points/lines", title="title", legend=True/False)

kind="bar"/"scatter"/"hist"/"box"/"density"/"area"/"pie"

x=x轴的colname, y=y轴的colname

plt.plot(x_data,y_data, label, color, marker, markerfacecolor, linestyle, linewidth)。x_data/y_data可以是list，也可以是pandas series; markerfacecolor是marker里面的填充颜色

plt.xlabel();plt.title(); plt.xlim(); plt.ylim(); plt.grid(True, linewidth, color, linestyle); plt.xticks()刻度线

plt.legend(loc, title, title_fontsize)。loc是图例位置，有upper left...; title是图例的标题; title_fontsize是标题大小

plt.axhline(y, c, ls,lw)。绘制水平参考线，y=竖坐标，c=color，ls=line style，lw=线宽
如果不创建subplot，plt.plot()/bar()/...函数会不停地在一张图里画

plt.savefig(path, dpi, facecolor)。facecolor是图表外图片底色，

1D plotting (就一列data还硬画)

df["colname"].plot(kind)

plt.plot([1000,2000,3000,4000])。跟上面不一样，一个是df调用，一个是plt调用

Hexagonal bin (散点太密集时用它)

数据sample: df = pd.DataFrame(np.random.randn(1000, 2), columns=["a", "b"]) df["b"] = df["b"] + np.arange(1000)

df.plot.hexbin(x="a", y="b", gridsize=25, C, reduce_C_function)

gridsize: 控制了x轴方向上的六边形大小，-默认值100。更大的size意味着更小更多的bins

每个六边形会计算范围内的散点，因此涉及到【计算什么值】和【怎么计算值】。C: 每个(x,y)被用于计算的value; reduce_C_function: 聚合后计算的方法，有mean, max, sum, std

模型情况，hexbin仅count周围散点的数量

Bar plot

有label非时间series->单列柱图: df.iloc[5].plot.bar()

dataframe->多列柱图(垂直): df.plot.bar(stacked)。stacked=True/False

dataframe->多列柱图(水平): df2.plot.barh(stacked)

plt.bar(参数跟之前的同理, width, align); width控制桌子的宽度; align控制柱子对其的位置，有edge/center

在一个figure中连续调用bar函数，需要用x-轴坐标+width来控制柱子的位置错开

Pie

数据sample : `series = pd.Series(3 * np.random.rand(4), index=["a", "b", "c", "d"], name="series")`

`series.plot.pie(figsize=(6, 6), legends, colors, autopct="%.2f", fontsize)`

饼状图适合方形：因此使用figsize或ax.set_aspect('equal')

单图：y参数指定一列

多图：subplots=True

legends：True/False决定要不要；None决定有没有文字；["AA", "BB", "CC", "DD"]

colors=["r", "g", "b", "c"]；autopct控制显示的数值格式

如果数据的和不足1，他们会被rescale

`plt.pie(data, labels, explode, autopct, wedgeprops)`。这个函数跟pandas的区别是-总和不足1时不会自动rescale

explode参数长度跟data，label一样，控制每个部分外移的距离；autopct决定label的显示格式"2.1f"；wedgeprops能让饼图变成环图，如{width: 0.3}，width设1即为饼图

2D plotting (终于正常一点)

`df.plot(kind=x,y,color)`

`plt.plot([1,2,3,4], [1000,2000,3000,4000], color, marker, linestyle)`。marker是点的颜色，linestyle是线的风格

`sns.countplot(df,x="Creditability")`

`sns.catplot(df,x='Creditability',y='Credit Amount',kind="violin")`

Box

`df.plot.box(column=["Col1", "Col2"], color=, color, sym="r+", vert=False, positions=[1, 4, 5, 6, 8], by="X")`。sym参数决定异常点形状，vertical=False让箱线图横过来，positions自定义箱线图距离

color = {"boxes": "DarkGreen", "whiskers": "DarkOrange", "medians": "DarkBlue", "caps": "Gray", }

by=["X","Y"]是指根据df的“X”这一列group，-然后分组绘制箱线图。每一个箱线图是一个group内每个列的箱线图

也可以先分组再画分组箱线图：`df_box.groupby("g").boxplot()`

`sns.boxplot(data=df["Credit Amount"])`

Area

数据sample : `df = pd.DataFrame(np.random.rand(10, 4), columns=["a", "b", "c", "d"])`

`df.plot.area(stacked)`

histogram

`df.plot.hist(stacked, bins, alpha, orientation="horizontal", cumulative=True)`

alpha是颜色透明度，bins是分桶数量，orientation决定了是横着的还是竖着的直方图，cumulative=True意味着不断累积，最后一个柱子的长度是1

`df.hist(figsize=(50,50))`也可以画

`plt.hist(y_data, bin_data, label)`

Scatter

数据sample : `df = pd.DataFrame(np.random.rand(50, 4), columns=["a", "b", "c", "d"])`
`df["species"] = pd.Categorical(["setosa"] * 20 + ["versicolor"] * 20 + ["virginica"] * 10)`

`df.plot.scatter(x="a", y="b", color, label, c)`

c="colname"，这个col会被用于cmap，决定每个散点的颜色。如果是类别变量则cmap会是个分段映射

s=size。控制每个散点的大小

画两组散点在一个图：`ax = df.plot.scatter(x="a", y="b", color="DarkBlue", label="Group 1")`
`df.plot.scatter(x="c", y="d", color="DarkGreen", label="Group 2", ax=ax);`

`plt.scatter(days, y_views, label, 其他的同理)`；

3D plotting

`df.plot.scatter(x,y,s=df["colname"])`

s=第三维度，如scatter图里是散点的大小

`sns.heatmap(df.corr())`



By cgeeeeh

cheatography.com/cgeeeeh/

Not published yet.

Last updated 4th December, 2023.

Page 2 of 2.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish

Yours!

<https://apollopadd.com>