import pandas as pd, import numpy as np, import math, import scipy.stats as st, import matplotlib.pyplot as plt

df = pd.read\_csv('Comune\_Bergamo\_-\_Incidenti\_stradali.csv') //apro csv

file.columns

lunghezza record: len(file) //in csv

file.info() // da info sui record e i loro tipi

file.get("Anno") //prendo record dell'attributo anno

//fa grafico a barre e lo stampa

file['Anno'].value\_counts(sort=False).plot.bar()

plt.show()

nferiti = file["N\_Feriti"].value\_counts() //freq assoluta

nferiti = file["N\_Feriti"].value\_counts(normalize=True) //freq relativa

n\_feriti.cumsum().plot() //ottimo per grafico di frequenze cumulative (usando freq relativa)

plt.show()

almeno un ... -> lavora con 1 - P(...)

file["N\_Feriti"].mean() //media di feriti -> valore atteso

file['N\_Feriti'].describe() //informazioni utilissime

per stima del parametro del modello uso le formule di valore atteso con attuale valore atteso

//modello geometrico

X = st.geom(p,loc=-1)

x = np.arange(11) #numero max di incidenti

plt.plot(x,X.pmf(x),'o') //funzione di massa di prob pmf

plt.show()

plt.scatter(df['N\_Feriti'],df['N\_Illesi'],color="brown") //scatterplot

plt.xlabel('Feriti') //nome a x

plt.ylabel('Illesi') //nome a y

plt.show()

df['N\_Feriti'].corr(df['N\_Illesi']) //indice di correlazione tra due vettori

per stimatore posso usare la media

df['N\_Illesi'].var() // per calcolare varianza

Z = st.norm() //crea normale

Z.ppf(0.995) //percent point function, inverso di cdf (cumulative disitribution function, ovvero fi)

df = pd.read\_csv('DATI-AMBIENTE.txt', sep=";", decimal=",", na\_values=' ') //separatore è importante

df["CO"].median() //mediana

medici.hist(bins=20) //istogramma con 20 colonne

plt.show()

from statsmodels.distributions.empirical\_distribution import ECDF //funzione cumulativa empirica

dist = ECDF(medici)

plt.plot(dist.x, dist.y)

plt.show()

l = 5

x = range(15)

p\_x = list(map(lambda \_: p\_poisson(\_, l), x))

plt.vlines(x, [0]\*len(x), p\_x)

plt.plot(x, p\_x, 'o')

plt.show() //a bastoncini

import scipy.stats as st

X = st.poisson(l)

X.pmf(x) //funzione di massa di probabilità

acqua['NomeSorgente'].unique() //ritorna i valori unici del campo nomesorgente del dataframe -> len = quanti valori unici

//gini

def gini(series):

return 1 - sum(series.value\_counts(normalize=True).map(lambda f: f\*\*2))

def normalized\_gini(series):

s = num\_values(series)

return s \* gini(series) / (s-1)

normalized\_gini(acqua['NomeSorgente'])

acqua['Oro'].value\_counts(normalize=True, sort=False)

plt.vlines(gold\_rel\_freq.index, 0, gold\_rel\_freq.values) //bastoncini

plt.plot(gold\_rel\_freq.index, gold\_rel\_freq.values, 'o')

plt.show()

media e mediana simili -> possibile poisson, inoltre in poisson //inoltre strettamente decrescente oppure con carattere di diminuizione non lineare allontanandosi dal massimo

import statsmodels.api as sm

sm.qqplot(acqua['DurezzaAcqua'], fit=True, line='45') //qqplot, se simile a bisettrice, allora è modello normale

plt.show()

acqua.plot.scatter('Oro', 'DurezzaAcqua') //scatter plot per valutare relazione tra due elementi

plt.show()

acqua['Oro'].corr(acqua['DurezzaAcqua']) //o anche indice di correlazione

acqua.groupby('NomeSorgente').mean() //groupby permette di dividere un dataframe in base a istanze dell'argomento del groupby

t = np.arange(0, 10, .1) //crea array

X = st.norm(200, 50)

x = np.linspace(0, 400, 50) //start, stop, numero di sample da generare

plt.plot(x, X.pdf(x))

plt.show()

dati.count()

len(dati) //teoricamente fanno la stessa cosa

plt.title(“titolo”) //per dare il titolo al grafico

dati.std() / dati.mean() //coefficiente di variazione

dati.plot.box() //plotbox / whiskers and box

plt.scatter(x, y)

plt.plot(x,y) //per collegare i puntini

plt.show()