**Class 1**

**import** pandas **as** pd

who **=** pd**.**read\_csv("WHO.csv")

who **=** pd**.**read\_csv("WHO.csv",parse\_dates**=**["Date\_reported"])

|  |  |
| --- | --- |
| Who | who**.**head() |
| type(who)  pandas.core.frame.DataFrame | type(who["Date\_reported"])  pandas.core.series.Series |
| who**.**ndim  2 | who["Date\_reported"]**.**ndim  1 |
| df**.**describe() |  |
| who**.**info() | who**.**Date\_reported **=** pd**.**to\_datetime(who["Date\_reported"])  0 2020-01-04  1 2020-01-05  2 2020-01-06  3 2020-01-07  4 2020-01-08  ...  31871 2020-07-31  31872 2020-07-31  31873 2020-07-31  31874 2020-07-31  31875 2020-07-31  Name: Date\_reported, Length: 31876, dtype: datetime64[ns] |
| who**.**Date\_reported**.**sort\_values() |
| who**.**shape  (31876, 6) | who**.**columns |
|  | poland **=** who[who**.**Country **==** "Poland"]  poland |

**Class 2**

|  |  |
| --- | --- |
| itay **=** who[who["Country"] **==** "Italy"]  **import** matplotlib.pyplot **as** plt  plt**.**plot(itay**.**Date\_reported,itay**.**Cumulative\_cases)  plt**.**show() | belaru **=** who[who**.**Country **==** "Belarus"]  **import** plotly.express **as** px  fig **=** px**.**line(belaru,"Date\_reported","New\_cases")  fig**.**update\_layout(hovermode**=**"x unified")  fig**.**show()  line\_chart("Belarus")  plt**.**plot(belaru**.**Date\_reported,belaru**.**New\_cases)  plt**.**show()    belaru**.**New\_cases**.**max()  1934  belaru**.**New\_cases**.**head() |
| **def** line\_chart(country):  c\_df **=** who[who["Country"]**==**country]  **if** country **==** "Italy":  plt**.**title(f"{country}:logistic convid commulaive cases")  **if** country **==** "India":  plt**.**title(f"{country}:Exponential convid commulaive cases")  **if** country **==** "Poland":  plt**.**title(f"{country}:Linear convid commulaive cases")  **if** country **==** "Italy":  plt**.**title(f"{country}:logrithmics convid commulaive cases")  plt**.**xlabel("Date")  plt**.**ylabel("Commulaive cases")  plt**.**ticklabel\_format(axis **=** "both")  plt**.**plot(c\_df**.**Date\_reported,c\_df**.**Cumulative\_cases)  plt**.**show() | plt**.**plot(uk**.**Date\_reported,uk**.**Cumulative\_cases,label **=** "The Uk")  plt**.**plot(France**.**Date\_reported,France**.**Cumulative\_cases,label **=** "The France")  plt**.**legend()  plt**.**show() |
|  | fig **=** px**.**line(France,"Date\_reported","New\_cases",title **=** "France:New Case", labels**=**{"Date\_reported":"Date","New\_cases":"New Cases"})  fig**.**update\_traces(mode**=**"markers+lines")  fig**.**update\_layout(hovermode**=**"x unified")  fig**.**show()  In [63]:  **import** plotly.graph\_objects **as** go  fig **=** go**.**Figure()    fig**.**add\_trace(go**.**Scatter(x **=** uk**.**Date\_reported,y **=** uk**.**Cumulative\_cases ,name **=** "UK"))  fig**.**add\_trace(go**.**Scatter(x **=** France**.**Date\_reported,y **=** France**.**Cumulative\_cases,name **=** "France"))  fig**.**show() |

**Class 3**

|  |
| --- |
| **Read url**  url **=** "https://archive.ics.uci.edu/static/public/275/bike+sharing+dataset.zip"  **import** requests  requests**=**requests**.**get(url) |
| **Zip write**  **with** open("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bike.zip","wb") **as** f:  f**.**write(requests**.**content) |
| **Z**ip extract  **import** zipfile  **with** zipfile**.**ZipFile("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bike.zip") **as** zipped:  zipped**.**extractall("C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bikeshared") |
| **Read csv**  **import** json  **import** pandas **as** pd  In [14]:  path **=** "C:\\Users\\hp\\Desktop\\ned\\Data Visualization\\bikeshared\\day.csv"  bike **=** pd**.**read\_csv(path) |
| bike['dteday'] **=** pd**.**to\_datetime(bike**.**dteday) |
| **import** matplotlib.pyplot **as** plt  In [24]:  plt**.**plot(bike**.**dteday,bike**.**cnt)  plt**.**show() |
| plt**.**plot(bike**.**dteday,bike**.**casual,label **=** "casual")  plt**.**plot(bike**.**dteday,bike**.**registered,label **=** "registered")  plt**.**show() |
| plt**.**scatter(bike**.**temp,bike**.**cnt)  plt**.**show() |
| bike["cnt"]**.**corr(bike["temp"]) |
| correlaion **=** bike**.**corr()  correlaion["workingday"]  instant -0.004337  season 0.012485  yr -0.002013  mnth -0.005901  holiday -0.253023  weekday 0.035790  workingday 1.000000  weathersit 0.061200  temp 0.052660  atemp 0.052182  hum 0.024327  windspeed -0.018796  casual -0.518044  registered 0.303907  cnt 0.061156  Name: workingday, dtype: float64 |
| correlaion["workingday"][['casual','registered']]  casual -0.518044  registered 0.303907  Name: workingday, dtype: float64 |

**Class 4**

|  |  |
| --- | --- |
| plt**.**scatter(bike**.**workingday,bike**.**casual)  plt**.**show() | bike[bike["workingday"] **==** 0]  pd**.**pivot\_table(bike,index **=** "workingday" ,values **=** ["registered","casual"],aggfunc**=**"mean") |
| bike**.**index  RangeIndex(start=0, stop=731, step=1) | bike[bike['workingday'] **==** 0][["registered","casual"]]**.**mean() |
| w\_mean['casual']**.**plot**.**bar()  plt**.**xticks(ticks **=** [0,1],labels **=** ["non-working","working"],rotation**=**0 )  plt**.**show() | w\_mean**=**pd**.**pivot\_table(bike,index **=** "workingday" ,values **=** ["registered","casual"],aggfunc**=**"mean")  w\_mean |
| bike['dteday']**.**dt**.**day  bike[['dteday','weekday']] | bike['dteday']**.**dt**.**strftime("%A %B")    bike**.**pivot\_table(index **=**'workingday' ,values **=** ["registered","casual"],aggfunc **=** 'mean')    week\_group **=** bike**.**groupby("weekday")  pd**.**options**.**display**.**max\_rows **=** **None**  week\_group**.**groups  week\_group**.**get\_group(1) |
| bike**.**pivot\_table(index **=**'workingday' ,values **=** ["registered","casual"],aggfunc **=** 'mean') | week **=** week\_group[['registered','casual']]**.**mean()  week |
| week['registered']**.**plot**.**bar()  plt**.**show() | bike**.**weathersit**.**unique()  (bike**.**weathersit**.**value\_counts(normalize**=**'True')**\***100)**.**plot**.**bar()  plt**.**show() |
| (bike**.**weathersit**.**value\_counts(normalize**=**'True')**\***100)**.**plot**.**pie()  plt**.**show() | bike**.**yr**.**map({0:2011,1:2012})  bike**.**yr**.**replace({0:2011,1:2012},inplace **=** **True**) |
| Bifurcated\_Data**=** bike**.**groupby("yr")["weathersit"]**.**value\_counts()  Bifurcated\_Data    Bifurcated\_Data[2012]**.**plot**.**bar() | Bifurcated\_Data[2011]    Bifurcated\_Data[2011]**.**plot**.**bar() |
| **import** seaborn **as** sns  bike["weathersit\_cat"] **=**bike**.**weathersit**.**map({1:"Clear",2:"Cloudy",3:"Rain"})  sns**.**countplot(data**=**bike,x**=**"yr",hue**=** "weathersit\_cat")  plt**.**show()  bike**.**cnt**.**unique()**.**size | sns**.**countplot(data**=**bike,x**=**"yr",hue**=** "weathersit")  plt**.**show()    bike**.**cnt**.**unique()**.**size  696  bike**.**cnt**.**value\_counts(bins**=**10) |
| bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index() | bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index()**.**plot**.**hist() |
| cnt **=** bike**.**cnt**.**value\_counts(bins**=**10)**.**sort\_index()  plt**.**hist(bike['cnt'])  plt**.**show() |  |

**Class 5**

|  |  |
| --- | --- |
| df[df["Salary"]**>**111] | df[["Salary","Age"]] |
| laptop**.**isnull()**.**sum() | laptop["year"]**.**fillna(2023,inplace**=True**)  laptop["date"] **=** pd**.**to\_datetime(laptop["date"],"%Y-%M-%D")  print(laptop["date"]) |
| **laptop.iloc[(rows),(columns)]**  laptop**.**iloc[500:600,1:2]  **for** i **in** laptop**.**columns:  print(i, ":" , laptop[i]**.**unique()) | group **=** laptop**.**groupby("TypeName")["Company"]  In [52]:  pd**.**DataFrame(group,columns **=**["TypeName","Company"]) |
| laptop["Weight"] **=** laptop["Weight"]**.**str**.**replace("kg","")  laptop["Weight"] **=** laptop["Weight"]**.**astype(float)  laptop["Weight"] | pd**.**DataFrame(laptop**.**groupby("TypeName")["Weight"]**.**sum()**.**reset\_index()) |
| ids **=** {'Apple':1, 'HP':2, 'Acer':3, 'Asus':4, 'Dell':5, 'Lenovo':6, 'Chuwi':7, 'MSI':8,  'Microsof':9, 'Toshiba':10, 'Huawei':11, 'Xiaomi':12, 'Vero':13, 'Razer':14,  'Mediacom':15, 'Samsung':16, 'Google':17, 'Fujitsu':18, 'LG':19}  print(ids) | laptop**.**rename(columns **=** {"Company":"New\_Compnay"},inplace**=** **True**)  laptop**.**drop(columns **=** {"Unnamed: 0"},inplace**=** **True**) |
| df1 **=** laptop[["New\_Compnay","TypeName"]]  df2 **=** laptop[["Inches","ScreenResolution"]]  pd**.**concat([df1,df2] , axis **=** 1)  df2**.**rename(columns **=**{"Inches":"New\_Compnay","ScreenResolution":"TypeName"},inplace**=** **True**)  pd**.**concat([df1,df2] , axis **=** 0) | **import** datetime **as** dt  laptop["date"]**.**dt**.**day |
| url **=** "https://archive.ics.uci.edu/static/public/483/behavior+of+the+urban+traffic+of+the+city+of+sao+paulo+in+brazil.zip"  **import** requests  responce **=** requests**.**get(url)  responce | **with** open("city.zip" ,"wb") **as** f:  f**.**write(responce**.**content)  In [95]:  pwd |
| **import** zipfile  In [105]:  **with** zipfile**.**ZipFile("city.zip") **as** zipped:  zipped**.**extractall("brazil\_city")  In [106]:  **import** pandas **as** pd  In [108]:  taffic **=** pd**.**read\_csv(r"C:\Users\hp\Desktop\ned\Data Visualization\brazil\_city\Behavior of the urban traffic of the city of Sao Paulo in Brazil\Behavior of the urban traffic of the city of Sao Paulo in Brazil.csv")  In [110]:  taffic**.**head() | taffic **=** pd**.**read\_csv(r"C:\Users\hp\Desktop\ned\Data Visualization\brazil\_city\Behavior of the urban traffic of the city of Sao Paulo in Brazil\Behavior of the urban traffic of the city of Sao Paulo in Brazil.csv",sep**=**';')  In [112]:  taffic**.**head()  taffic**.**isnull()**.**sum() |
| taffic**.**index  RangeIndex(start=0, stop=135, step=1) | taffic**.**columns |
| taffic**.**isnull()**.**sum(axis **=** "index") | taffic**.**isnull()**.**sum(axis **=** "columns") |
| taffic**.**isnull()**.**sum(axis **=** 1) | taffic["Slowness in traffic (%)"]**.**str**.**replace(",",".",inplace **=True**)  Error  taffic["Slowness in traffic (%)"] **=** taffic["Slowness in  traffic(%)"]**.**str**.**replace(",",".")**.**astype(float)  taffic["Slowness in traffic (%)"] |
| taffic**.**dtypes | taffic["Slowness in traffic (%)"]**.**value\_counts()    taffic["Slowness in traffic (%)"]**.**value\_counts(bins**=**10) |
| taffic["Slowness in traffic (%)"]**.**value\_counts(bins**=**10)**.**sort\_index() | taffic["Slowness in traffic (%)"]**.**plot**.**hist() |
| **import** matplotlib.pyplot **as** plt  plt**.**hist(taffic["Slowness in traffic (%)"])  plt**.**show() | incidence **=** taffic**.**drop(['Hour (Coded)','Slowness in traffic (%)'] ,axis **=**1)**.**copy() *# axis column*  incidence |
| incidence**.**sum()    incidence**.**sum()**.**sum()   numeriac values -->scatter plotcategorical and numeric -->barplotcontinious values -->histogrmtime against value -->line chart | incidence**.**sum()**.**plot**.**barh()    taffic**.**corr()['Slowness in traffic (%)']**.**drop("Hour (Coded)")**.**abs()    taffic**.**corr()['Slowness in traffic (%)']**.**drop("Hour (Coded)")**.**sort\_values(ascending **=** **False**) |
| plt**.**scatter(taffic["Slowness in traffic (%)"],taffic["Lack of electricity"]) | plt**.**scatter(taffic["Slowness in traffic (%)"],taffic["Point of flooding"]) |

**Class 6**

|  |
| --- |
| df["Slowness in traffic (%)"] **=** df["Slowness in traffic(%)"]**.**str**.**replace(",",".")**.**astype(float) |
| df\_20 **=** df[df["Slowness in traffic (%)"] **>**20] |
| df\_20**.**drop(["Slowness in traffic (%)","Hour (Coded)"],axis **=** 1) |
| df\_20**.**drop(["Slowness in traffic (%)","Hour (Coded)"],axis **=** 1)**.**sum()**.**sort\_values(ascending **=** **False**) |
| iloc- Last loaction not includedloc - last location is included df.loc[:27]  df[["Hour (Coded)","Immobilized bus"]] |
| for i in range(0,len(df),27):  print(i)  for i in zip(["waqar","waqas","Amara"],[1,2,3]):  print(i)  ('waqar', 1)  ('waqas', 2)  ('Amara', 3)  for i,j in zip(["waqar","waqas","Amara"],[1,2,3]):  print(i)  waqar  waqas  Amara  days = ["Mon","Tue","Wed","Thu","Fri"]  traffic\_per\_day ={}  for day,j in zip (days,range(0,len(df),27)):  print(day,j)  Mon 0  Tue 27  Wed 54  Thu 81  Fri 108  days = ["Mon","Tue","Wed","Thu","Fri"]  traffic\_per\_day ={}  for day,i in zip(days,range(0,len(df),27)):  daily\_data = df.iloc[i:i+27]    traffic\_per\_day[day] = daily\_data |
| for df in traffic\_per\_day.values():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.show()    for day,df in traffic\_per\_day.items():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"],label =day)  plt.legend()  plt.ylim(0,25)  plt.show() |
| plt.subplot(3,2,2)  plt.subplot(3,2,3)  plt.subplot(3,2,6)  plt.show()    plt.figure(figsize=(10,12))  plt.subplot(3,2,2)  plt.subplot(3,2,3)  plt.subplot(3,2,6)  plt.show()    plt.figure(figsize=(10,12))  for i,df in zip(range(0,5), traffic\_per\_day.values()):  plt.subplot(3,2,i+1)  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.title(days[i])  plt.ylim(0,25)  plt.show()      plt.figure(figsize=(10,12))  for i,df in zip(range(0,5), traffic\_per\_day.values()):  plt.subplot(3,2,i+1)  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"])  plt.title(days[i])  plt.ylim(0,25)    plt.subplot(3,2,6)  for day,df in traffic\_per\_day.items():  plt.plot(df["Hour (Coded)"],df["Slowness in traffic (%)"],label =day)  plt.legend()  plt.ylim(0,25)  plt.title("comparision")  plt.savefig("traffic\_variation.png")  plt.show() |
| pro **=** pd**.**read\_csv("property.csv")  In [69]: pro**.**head()  **import** seaborn **as** sns  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice')  plt**.**show() |
| sns**.**set\_theme()  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn")  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300))  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Rooms')  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Rooms',sizes**=**[200,30])  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms")  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",markers **=**["^","v"] )  plt**.**show() |
| sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",col **=** "Year" )  plt**.**show() |
| pro["Land Slope"]**.**unique()  sns**.**relplot(data**=**pro,x**=**'Gr Liv Area',y**=**'SalePrice',  hue**=**'Overall Qual',palette**=**"RdYlGn",  size**=** 'Garage Area',sizes**=**(1,300)  ,style**=**"Rooms",col **=** "Year", row **=** "Land Slope")  plt**.**show() |

Class 7

|  |
| --- |
| url **=** "https://archive.ics.uci.edu/static/public/492/metro+interstate+traffic+volume.zip"  **import** requests  responce **=** requests**.**get(url) |
| **with** open("Metro Interstate Traffic Volume.zip","wb") **as** f:  f**.**write(responce**.**content) |
| **import** zipfile  **with** zipfile**.**ZipFile("Metro Interstate Traffic Volume.zip") **as** zipped:  zipped**.**extractall("Metro Interstate Traffic Volume") |
| **import** pandas **as** pd  i\_94 **=** pd**.**read\_csv(r"Metro\_Interstate\_Traffic\_Volume.csv") |
| new\_df **=** i\_94[**~**i\_94["holiday"]**.**str**.**contains("None")]  new\_df |
| **import** matplotlib.pyplot **as** plt  i\_94**.**traffic\_volume**.**plot**.**hist()  plt**.**show() |
| i\_94**.**date\_time **=** pd**.**to\_datetime(i\_94**.**date\_time)  i\_94**.**date\_time**.**dtype |
| day\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 7 ) **&** (i\_94**.**date\_time**.**dt**.**hour **<** 19 )  day **=** i\_94[day\_bol]  day    day**.**date\_time**.**max()  Timestamp('2018-09-30 18:00:00')  day**.**date\_time**.**dt**.**time    In [36]:  day**.**date\_time**.**dt**.**time**.**min()  Out[37]:  datetime.time(7, 0)  In [38]:  day**.**date\_time**.**dt**.**time**.**max()  Out[38]:  datetime.time(18, 0)  In [39]:  night\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 19 ) **|** (i\_94**.**date\_time**.**dt**.**hour **<** 7 )  night**=**i\_94[night\_bol]  In [40]:  night**.**date\_time**.**dt**.**hour**.**unique()  Out[44]:  array([19, 20, 21, 22, 23, 0, 1, 2, 3, 4, 5, 6], dtype=int64)  In [45]:  day**.**date\_time**.**dt**.**hour**.**unique()  Out[45]:  array([ 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 8, 7], dtype=int64) |
| plt**.**figure(figsize**=**(11,4))  plt**.**subplot(1,2,1)  plt**.**hist(day["traffic\_volume"])  plt**.**title("traffic volumne :day")  plt**.**xlim(0,8000)  plt**.**ylim(0,8500)  plt**.**xlabel("traffic volume")  plt**.**ylabel("number of incident")  plt**.**subplot(1,2,2)  plt**.**hist(night["traffic\_volume"])  plt**.**title("traffic volumne :night")  plt**.**xlim(0,8000)  plt**.**ylim(0,8500)  plt**.**xlabel("traffic volume")  plt**.**show() |
| night**.**traffic\_volume**.**value\_counts(bins**=**10)**.**sort\_index() |
| day["month"] **=**day**.**date\_time**.**dt**.**strftime("%B")  day |
| group **=** day**.**groupby("month")  by\_month **=** group["traffic\_volume"]**.**mean()  by\_month |
| by\_month**.**plot**.**line()  plt**.**show() |
| day["year"] **=** day**.**date\_time**.**dt**.**year  only\_july **=** day[day["month"]**==**"July"]  yearly\_group **=** only\_july**.**groupby("year")["traffic\_volume"]**.**mean()  yearly\_group    only\_july**.**groupby("year")["traffic\_volume"]**.**mean()**.**plot**.**line()  plt**.**show() |

Class 8

|  |
| --- |
| **import** pandas **as** pd  i\_94 **=** pd**.**read\_csv("Metro\_Interstate\_Traffic\_Volume.csv")  i\_94**.**date\_time **=** pd**.**to\_datetime(i\_94**.**date\_time)  i\_94**.**date\_time**.**dt**.**strftime("%A") |
| avrageDayofWeek **=** i\_94**.**groupby("dayofweek")["traffic\_volume"]**.**mean()  avrageDayofWeek    avrageDayofWeek**.**plot**.**line()    avrageDayofWeek**.**plot(kind**=**"line") |
| i\_94["hours"] **=** i\_94**.**date\_time**.**dt**.**hour  day\_bol **=** (i\_94**.**date\_time**.**dt**.**hour **>=** 7) **&** (i\_94**.**date\_time**.**dt**.**hour **<**19)  day **=** i\_94[day\_bol]  day  day["dayinwords"] **=** day**.**date\_time**.**dt**.**strftime("%A") |
| avrageDayofWeek **=** day**.**groupby("dayofweek")["traffic\_volume"]**.**mean()  avrageDayofWeek  avrageDayofWeek**.**plot**.**line() |
| weekend **=** day[**~**(day["dayofweek"]**<=**4)]  weekend  weekdays**.**pivot\_table(index**=**"hours",values**=**"traffic\_volume",) |
| **import** matplotlib.pyplot **as** plt  plt**.**figure(figsize**=**(11,4))  plt**.**subplot(1,2,1)  Hourly\_weekday\_avg **=** weekdays**.**pivot\_table(index**=**"hours",values **=** "traffic\_volume")  plt**.**plot(Hourly\_weekday\_avg**.**index,Hourly\_weekday\_avg)  plt**.**ylim(1500,6500)  plt**.**subplot(1,2,2)  Hourly\_weekend\_avg **=** weekend**.**pivot\_table(index**=**"hours",values **=** "traffic\_volume")  plt**.**plot(Hourly\_weekend\_avg**.**index,Hourly\_weekend\_avg)  plt**.**ylim(1500,6500)  plt**.**show() |
| day**.**select\_dtypes(include**=**"number")    day**.**select\_dtypes(include**=**"number")**.**corr()["traffic\_volume"]    plt**.**scatter(day["temp"],day**.**traffic\_volume)    day**.**groupby("weather\_main")["traffic\_volume"]**.**mean()    day**.**groupby("weather\_main")["traffic\_volume"]**.**mean()**.**plot**.**barh()  plt**.**figure(figsize**=**(6,11))  day**.**groupby(["weather\_main","weather\_description"])["traffic\_volume"]**.**mean()**.**plot**.**barh()    plt**.**figure(figsize**=**(6,11))  day**.**groupby(["weather\_main","weather\_description"])["traffic\_volume"]**.**mean()**.**plot**.**barh()    plt**.**figure(figsize**=**(5,8))  day**.**groupby("weather\_description")["traffic\_volume"]**.**mean()**.**plot**.**barh()    day["weather\_description"]**.**value\_counts()**.**sort\_values(ascending**=** **False**)    day["weather\_description"] **=** day["weather\_description"]**.**str**.**replace("Sky is Clear","sky is clear")  day["weather\_description"]**.**value\_counts()**.**sort\_values(ascending**=** **False**) |

Class 9

|  |  |
| --- | --- |
| **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  In [3]:  death **=** pd**.**read\_csv("deaths.csv")  death | |
| **design princilple.. gendrate the disign option.. choose among those option**   * data element * structral elements * decoration   **-->familiarty principle**  **-->maximizig the data ink ratio =(data ink)/(data +sturutal + decotion)**  **sturutal + decotion== non dat ink**  **most customization offered by matplotlib**  **color of company logo make that color of graph** | |
| plt**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() | *#oo inteface*  fig,ax **=** plt**.**subplots()*#oo interface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() |
| *#oo interface*  fig,ax **=** plt**.**subplots(figsize**=**(4.5,6)) *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  plt**.**show() | *#oo inteface remove tick*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  plt**.**show() |
| **for** spines **in** ax**.**spines:  print(spines)  *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  plt**.**show()   |  | | --- | |  | | *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  plt**.**title("The death trol world wide is 1.5M+")  plt**.**show() |
| *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45)*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**set\_xticklabels(["0","150,000","300,000"])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() |
| *#oo inteface remove spines*  fig,ax **=** plt**.**subplots() *#oo iterface*  ax**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax**.**set\_xticks([0,150000,300000])  ax**.**set\_xticklabels(["0","150,000","300,000"])  ax**.**set\_yticklabels([])  ax**.**xaxis**.**tick\_top()  ax**.**tick\_params(top**=False**)  ax**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  **for** i,c **in** enumerate(death**.**Country\_Other):  ax**.**text(x**=-**80000,y**=**i**-**0.15,s**=**c)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | |
| *#oo inteface remove spines*  fig,(ax1,ax2) **=** plt**.**subplots(nrows**=**1,ncols**=**2,figsize**=**(15 , 6)) *#oo iterface*  ax1**.**barh(death**.**Country\_Other,death**.**Total\_Deaths)*#functinal interface*  ax2**.**barh(death**.**Country\_Other,death**.**Total\_Deaths,height **=** 0.45 )*#functinal interface*  ax2**.**tick\_params(bottom**=False**,left **=** **False**)  **for** spines\_val **in** ax2**.**spines**.**values():  spines\_val**.**set\_visible(**False**)  ax2**.**set\_xticks([0,150000,300000])  ax2**.**set\_xticklabels(["0","150,000","300,000"])  ax2**.**set\_yticklabels([])  ax2**.**xaxis**.**tick\_top()  ax2**.**tick\_params(top**=False**)  ax2**.**text(x**=-**80000,y**=**23.5,s**=**"The death trol world wide is 1.5M+",size**=**17,weight**=** "bold")  ax2**.**text(x**=-**80000,y**=**22.5,s**=**"Top 20 Countires by death Tool(December 2020)",size**=**12)  **for** i,c **in** enumerate(death**.**Country\_Other):  ax2**.**text(x**=-**80000,y**=**i**-**0.15,s**=**c)  ax2**.**axvline(150000,ymin**=**0.045,ymax**=**1,alpha **=** 0.5,c**=**"grey")  plt**.**subplots\_adjust(wspace**=**0.5)  *#plt.title("The death trol world wide is 1.5M+")*  plt**.**show() | |

Class 10

|  |
| --- |
| **import** pandas **as** pd  **import** matplotlib.pyplot **as** plt  In [45]:  death **=** pd**.**read\_csv("avg\_deaths.csv")  death |
| **import** calendar  calendar**.**monthrange(2020,1)  Out[49]:  (2, 31)  In [48]:  \_,num\_day **=**calendar**.**monthrange(2020,1)  In [49]:  num\_day  Out[49]:  31 |
| death**.**columns **=** ['Year','Month','Num\_days' ,'New\_deaths', ]  death |
| **def** days(year,month):  \_,num\_days **=** calendar**.**monthrange(year,month)  **return** num\_days  days **=**death[["Year","Month"]]**.**apply(**lambda** row:days(row["Year"],row["Month"]) ,axis**=**1)  days |
| death**.**loc[11,"Num\_days"]**//=**2 |
| death["Monthly\_deaths"] **=** death**.**Num\_days**.**mul(death**.**New\_deaths) |
| death**.**rename({"New\_deaths":"Avg\_deaths"}, axis**=**1, inplace**=True**) |
| death["Cum\_deaths"] **=** death**.**Monthly\_deaths**.**cumsum() |
| fig,(ax1,ax2,ax3,ax4 ) **=** plt**.**subplots(nrows **=** 4,ncols **=** 1,figsize**=**(6,8))  axes **=** [ax1,ax2,ax3,ax4]  **for** ax **in** axes:  ax**.**plot(death["Month"],death["Avg\_deaths"],alpha**=**0.1)  ax**.**set\_xticklabels([])  ax**.**set\_yticklabels([])  ax**.**tick\_params(bottom **=False**,left**=False**)  **for** spines **in** ax**.**spines:  ax**.**spines[spines]**.**set\_visible(**False**)  ax1**.**plot(death["Month"][:3],death["Avg\_deaths"][:3])  ax1**.**text(x**=**0.5,y**=-**80,s**=**"0",alpha **=**0.5)  ax1**.**text(x**=**3.5,y**=** 2000,s**=**"1844",alpha **=**0.5)  ax1**.**text(x**=**11.5,y**=** 2400,s**=**"2247",alpha **=**0.5)  ax1**.**text(x**=**1.5,y**=** **-**300,s**=**"Jan-Mar",rotation**=**3)  ax1**.**text(0.5,3500,"The virius kill 900 people every day",size**=**14, weight **=** "bold")  ax1**.**text(0.5,3100,"The Avarge number of daily death" ,size**=**12)  ax2**.**plot(death["Month"][2:6],death["Avg\_deaths"][2:6])  ax2**.**text(x**=**3.7,y**=** 800,s**=**"Mar-Jun")  ax3**.**plot(death["Month"][5:10],death["Avg\_deaths"][5:10])  ax3**.**text(x**=**7.1,y**=** 500,s**=**"Jun-Oct")  ax4**.**plot(death["Month"][9:],death["Avg\_deaths"][9:])  ax4**.**text(x**=**10.5,y**=** 660,s**=**"Oct-Dec",rotation**=**45)  **for** ax **in** axes:  ax**.**axhline(y**=**2000,xmin**=**0.5,xmax**=**0.8,c**=**"blue",linewidth**=**6,alpha**=** 0.5)  plt**.**show() |
| cum\_cases **=** [death**.**loc[2,"Cum\_deaths"],death**.**loc[5,"Cum\_deaths"],death**.**loc[9,"Cum\_deaths"],death**.**loc[11,"Cum\_deaths"]  ]  In [97]:  cum\_cases  Out[97]:  [2387, 126145, 227055, 297510]  In [105]:  propotion **=** [round(i**/**cum\_cases[**-**1],2) **for** i **in** cum\_cases]  propotion  Out[105]:  [0.01, 0.42, 0.76, 1.0]  In [110]:  propotion **=** []  **for** i **in** cum\_cases:  v **=** i**/**297510  final **=** round(v,2)  propotion**.**append(final)  In [111]:  propotion  Out[111]:  [0.01, 0.42, 0.76, 1.0] |

Class 11

|  |
| --- |
| import calendar  calendar.monthrange(2020,1)  (2, 31)  \_,num\_day =calendar.monthrange(2020,1)  num\_day  31 |
| def days(year,month):  \_,num\_days = calendar.monthrange(year,month)  return num\_days  death.iloc[0]  Year 2020  Month 1  Num\_days 0  New\_deaths 0  Name: 0, dtype: int64 |
| days = death[["Year","Month"]].apply(lambda row:days(row["Year"],row["Month"]) ,axis=1)  days  0 31  1 29  2 31  3 30  4 31  5 30  6 31  7 31  8 30  9 31  10 30  11 31  dtype: int32 |
| death.Num\_days = days  death.loc[11,"Num\_days"]//=2  death   |  | **Year** | **Month** | **Num\_days** | **New\_deaths** | | --- | --- | --- | --- | --- | | **0** | 2020 | 1 | 31 | 0 | | **1** | 2020 | 2 | 29 | 0 | | **2** | 2020 | 3 | 31 | 77 | | **3** | 2020 | 4 | 30 | 1844 | | **4** | 2020 | 5 | 31 | 1448 | | **5** | 2020 | 6 | 30 | 785 | | **6** | 2020 | 7 | 31 | 769 | | **7** | 2020 | 8 | 31 | 1020 | | **8** | 2020 | 9 | 30 | 739 | | **9** | 2020 | 10 | 31 | 751 | | **10** | 2020 | 11 | 30 | 1225 | | **11** | 2020 | 12 | 15 | 2247 | |
| death["Monthly\_deaths"] = death.Num\_days.mul(death.New\_deaths)  death.rename({"New\_deaths":"Avg\_deaths"}, axis=1, inplace=True)  death   |  | **Year** | **Month** | **Num\_days** | **Avg\_deaths** | **Monthly\_deaths** | | --- | --- | --- | --- | --- | --- | | **0** | 2020 | 1 | 31 | 0 | 0 | | **1** | 2020 | 2 | 29 | 0 | 0 | | **2** | 2020 | 3 | 31 | 77 | 2387 | | **3** | 2020 | 4 | 30 | 1844 | 55320 | | **4** | 2020 | 5 | 31 | 1448 | 44888 | | **5** | 2020 | 6 | 30 | 785 | 23550 | | **6** | 2020 | 7 | 31 | 769 | 23839 | | **7** | 2020 | 8 | 31 | 1020 | 31620 | | **8** | 2020 | 9 | 30 | 739 | 22170 | | **9** | 2020 | 10 | 31 | 751 | 23281 | | **10** | 2020 | 11 | 30 | 1225 | 36750 | | **11** | 2020 | 12 | 15 | 2247 | 33705 | |
| cum\_cases = [death.loc[2,"Cum\_deaths"],death.loc[5,"Cum\_deaths"],death.loc[9,"Cum\_deaths"],death.loc[11,"Cum\_deaths"]  ]  propotion = [round(i/cum\_cases[-1],2) for i in cum\_cases]  propotion  [0.01, 0.42, 0.76, 1.0] |
| xmaxs = [0.5+ i\*(0.8-0.5) for i in propotion]  xmaxs  [0.503, 0.626, 0.728, 0.8]  fig,(ax1,ax2,ax3,ax4 ) = plt.subplots(nrows = 4,ncols = 1,figsize=(6,8))  axes = [ax1,ax2,ax3,ax4]  for ax in axes:  ax.plot(death["Month"],death["Avg\_deaths"],alpha=0.1)  ax.set\_xticklabels([])  ax.set\_yticklabels([])  ax.tick\_params(bottom =False,left=False)  for spines in ax.spines:  ax.spines[spines].set\_visible(False)  ax1.plot(death["Month"][:3],death["Avg\_deaths"][:3])  ax1.text(x=0.5,y=-80,s="0",alpha =0.5)  ax1.text(x=3.5,y= 2000,s="1844",alpha =0.5)  ax1.text(x=11.5,y= 2400,s="2247",alpha =0.5)  ax1.text(x=1.5,y= -300,s="Jan-Mar",rotation=3)  ax1.text(0.5,3500,"The virius kill 900 people every day",size=14, weight = "bold")  ax1.text(0.5,3100,"The Avarge number of daily death" ,size=12)  ax2.plot(death["Month"][2:6],death["Avg\_deaths"][2:6])  ax2.text(x=3.7,y= 800,s="Mar-Jun")  ax3.plot(death["Month"][5:10],death["Avg\_deaths"][5:10])  ax3.text(x=7.1,y= 500,s="Jun-Oct")  ax4.plot(death["Month"][9:],death["Avg\_deaths"][9:])  ax4.text(x=10.5,y= 660,s="Oct-Dec",rotation=45)  for ax,deaths, xmax in zip(axes,cum\_cases,xmaxs):  ax.axhline(y=2000,xmin=0.5,xmax=0.8,c="blue",linewidth=6,alpha= 0.5)  ax.axhline(y=2000,xmin=0.5,xmax=xmax,c="blue",linewidth=6)  ax.text(x=7.5,y=2200,s=format(deaths,","),weight="bold")  plt.show() |

|  |
| --- |
| url = "https://archive.ics.uci.edu/static/public/186/wine+quality.zip"  import requests  import pandas as pd  responce = requests.get(url,verify=False)  with open("wine.zip","wb") as f:  f.write(responce.content) |
| import zipfile  with zipfile.ZipFile("wine.zip") as Zipped:  Zipped.extractall("wine")  red\_wine = pd.read\_csv("winequality-red.csv" ,sep = ";")  white\_wine = pd.read\_csv("winequality-white.csv",sep = ";") |
| red\_corr = red\_wine.corr()["quality"][:-1] # red\_corr.drop["quality"]  white\_corr = white\_wine.corr()["quality"][:-1] |
| import matplotlib.pyplot as plt  import matplotlib.style as style  print(style.available)  ['Solarize\_Light2', '\_classic\_test\_patch', '\_mpl-gallery', '\_mpl-gallery-nogrid', 'bmh', 'classic', 'dark\_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn-v0\_8', 'seaborn-v0\_8-bright', 'seaborn-v0\_8-colorblind', 'seaborn-v0\_8-dark', 'seaborn-v0\_8-dark-palette', 'seaborn-v0\_8-darkgrid', 'seaborn-v0\_8-deep', 'seaborn-v0\_8-muted', 'seaborn-v0\_8-notebook', 'seaborn-v0\_8-paper', 'seaborn-v0\_8-pastel', 'seaborn-v0\_8-poster', 'seaborn-v0\_8-talk', 'seaborn-v0\_8-ticks', 'seaborn-v0\_8-white', 'seaborn-v0\_8-whitegrid', 'tableau-colorblind10']  style.use("ggplot") |
| fig , ax = plt.subplots(figsize=(9,5))  ax.barh(white\_corr.index,white\_corr,label="white",left=2)  ax.barh(red\_corr.index,red\_corr.values,label="red")  ax.grid(visible = False)  ax.set\_xticklabels([])  ax.set\_yticklabels([])  ax.set\_xticks([])  ax.set\_yticks([])  plt.show() |
| import coords  dir(coords)  ['\_\_builtins\_\_',  '\_\_cached\_\_',  '\_\_doc\_\_',  '\_\_file\_\_',  '\_\_loader\_\_',  '\_\_name\_\_',  '\_\_package\_\_',  '\_\_spec\_\_',  'x\_coords',  'y\_coord'] |
| xcoords = coords.x\_coords  ycoords = coords.y\_coord  print(xcoords)  print(ycoords)  {'Alcohol': 0.82, 'Sulphates': 0.77, 'pH': 0.91, 'Density': 0.8, 'Total Sulfur Dioxide': 0.59, 'Free Sulfur Dioxide': 0.6, 'Chlorides': 0.77, 'Residual Sugar': 0.67, 'Citric Acid': 0.76, 'Volatile Acidity': 0.67, 'Fixed Acidity': 0.71}  9.8 |
| fig , ax = plt.subplots(figsize=(9,5))  ax.barh(white\_corr.index,white\_corr,label="white",left=2)  ax.barh(red\_corr.index,red\_corr.values,label="red")  ax.grid(visible = False)  ax.set\_xticklabels([])  ax.set\_yticklabels([])  ax.set\_xticks([])  ax.set\_yticks([])  xcoords = coords.x\_coords  ycoords = coords.y\_coord  for label,x\_cord in xcoords.items():  ax.text(x=x\_cord,y=ycoords,s=label)  ycoords-=1  ax.axvline(0.5,c="grey",alpha=0.1,linewidth=1,ymin=0.1,ymax=0.9)  ax.axvline(1.45,c="grey",alpha=0.1,linewidth=1,ymin=0.1,ymax=0.9)  ax.axhline(-1,xmin=0.01,xmax=0.32,c='grey',linewidth=1,alpha=0.5)  ax.axhline(-1,xmin=0.67,xmax=0.98,c='grey',linewidth=1,alpha=0.5)  ax.text(x=-0.5,y=-1.7,s="-0.5"+" "\*37+"+0.5" ,c="grey",alpha=0.5)  ax.text(x=1.5,y=-1.7,s="-0.5"+" "\*37+"+0.5" ,c="grey",alpha=0.5)  plt.show() |

Class -12

|  |
| --- |
| possitive\_red = red\_corr>0  possitive\_white = white\_corr>0  color\_red = possitive\_red.map({True:"#3cb371",False:"#ff0000"} )  color\_white = possitive\_white.map({True:"#3cb371",False:"#ff0000"} ) |
| import matplotlib.pyplot as plt  import matplotlib.style as style  import coords  style.use("ggplot") |
| fig , ax = plt.subplots(figsize=(9,5))  ax.barh(white\_corr.index,white\_corr,label="white",left=2,color=color\_white)  ax.barh(red\_corr.index,red\_corr.values,label="red",color=color\_red)  ax.grid(visible = False)  ax.set\_xticklabels([])  ax.set\_yticklabels([])  ax.set\_xticks([])  ax.set\_yticks([])  xcoords = coords.x\_coords  ycoords = coords.y\_coord  for label,x\_cord in xcoords.items():  ax.text(x=x\_cord,y=ycoords,s=label)  ycoords-=1  ax.axvline(0.5,c="grey",alpha=0.1,linewidth=1,ymin=0.1,ymax=0.9)  ax.axvline(1.45,c="grey",alpha=0.1,linewidth=1,ymin=0.1,ymax=0.9)  ax.axhline(-1,xmin=0.01,xmax=0.32,c='grey',linewidth=1,alpha=0.5)  ax.axhline(-1,xmin=0.67,xmax=0.98,c='grey',linewidth=1,alpha=0.5)  ax.text(x=-0.5,y=-1.7,s="-0.5"+" "\*37+"+0.5" ,c="grey",alpha=0.5)  ax.text(x=1.5,y=-1.7,s="-0.5"+" "\*37+"+0.5" ,c="grey",alpha=0.5)  ax.axhline(11,xmin=0.01,xmax=0.32,c='grey',linewidth=1,alpha=0.5)  ax.axhline(11,xmin=0.67,xmax=0.98,c='grey',linewidth=1,alpha=0.5)  ax.text(x=-0.33,y=11.2,s="RED WINE",weight = "bold")  ax.text(x=1.75,y=11.2,s="WHITE WINE",weight = "bold")  ax.text(x=-0.7,y=13.5,s="Wine quality most storongly coorelted with Alchol",weight = "bold",size=17)  ax.text(x=-0.7,y=12.8,s="Coralaion with the wine quality and its properties",size=13)  ax.text(x=-0.7,y=-2.7,s="©Digitech Senegy Pvt Limited"+" "\*68+"Ned university",size=12, backgroundcolor="#4d4d4d",color="#f0f0f0")  plt.show() |
| euro = pd.read\_csv("euro.csv")  euro\_dollar = euro[[r"Period\Unit:",r"[US dollar ]"]].copy()  euro\_dollar   |  | **Period\Unit:** | **[US dollar ]** | | --- | --- | --- | | **0** | 2021-01-08 | 1.2250 | | **1** | 2021-01-07 | 1.2276 | | **2** | 2021-01-06 | 1.2338 | | **3** | 2021-01-05 | 1.2271 | | **4** | 2021-01-04 | 1.2296 | | **...** | ... | ... | | **5694** | 1999-01-08 | 1.1659 | | **5695** | 1999-01-07 | 1.1632 | | **5696** | 1999-01-06 | 1.1743 | | **5697** | 1999-01-05 | 1.1790 | | **5698** | 1999-01-04 | 1.1789 |   5699 rows × 2 columns |
| euro\_dollar.rename(columns={r"Period\Unit:":"Date",r"[US dollar ]":"USD"},inplace =True) |
| euro\_dollar.Date = pd.to\_datetime(euro\_dollar.Date)  euro\_dollar   |  | **Date** | **USD** | | --- | --- | --- | | **0** | 2021-01-08 | 1.2250 | | **1** | 2021-01-07 | 1.2276 | | **2** | 2021-01-06 | 1.2338 | | **3** | 2021-01-05 | 1.2271 | | **4** | 2021-01-04 | 1.2296 | | **...** | ... | ... | | **5694** | 1999-01-08 | 1.1659 | | **5695** | 1999-01-07 | 1.1632 | | **5696** | 1999-01-06 | 1.1743 | | **5697** | 1999-01-05 | 1.1790 | | **5698** | 1999-01-04 | 1.1789 | |
| euro\_dollar.sort\_values(by="Date",inplace=True)  euro\_dollar.reset\_index(drop=True,inplace=True)  euro\_dollar   |  | **Date** | **USD** | | --- | --- | --- | | **0** | 1999-01-04 | 1.1789 | | **1** | 1999-01-05 | 1.1790 | | **2** | 1999-01-06 | 1.1743 | | **3** | 1999-01-07 | 1.1632 | | **4** | 1999-01-08 | 1.1659 | | **...** | ... | ... | | **5694** | 2021-01-04 | 1.2296 | | **5695** | 2021-01-05 | 1.2271 | | **5696** | 2021-01-06 | 1.2338 | | **5697** | 2021-01-07 | 1.2276 | | **5698** | 2021-01-08 | 1.2250 | |
| euro\_dollar["USD"].value\_counts()  - 62  1.2276 9  1.1215 8  1.1305 7  1.1797 6  ..  1.2571 1  1.2610 1  1.2651 1  1.2632 1  1.2193 1  Name: USD, Length: 3528, dtype: int64  bol = euro\_dollar['USD'] != "-"  euro\_dollar = euro\_dollar[bol]  euro\_dollar.USD = euro\_dollar.USD.astype(float) |
| plt.plot(euro\_dollar["Date"],euro\_dollar.USD)  plt.show() |
| moving avarge ,rolling avarage -->moving window rolloing window Moving Windows 7 Day  30 Day  50 Day  100 Day  365 Day  euro\_dollar.USD.rolling(7).mean().head(50)  rolling = [7,30,50,100,365]  plots = [2,3,4,5,6]  plt.figure(figsize=(11,9))  plt.subplot(3,2,1)  plt.plot(euro\_dollar["Date"],euro\_dollar.USD)  plt.title("Orginal")  for rol,plot in zip(rolling,plots):  plt.subplot(3,2,plot)  plt.plot(euro\_dollar["Date"],euro\_dollar.USD.rolling(rol).mean())  plt.title(f"Moving Avarge:{rol}")  plt.tight\_layout()  plt.show() |
| euro\_dollar["Rolling\_mean\_30"]=euro\_dollar.USD.rolling(30).mean()  euro\_dollar   |  | **Date** | **USD** | **Rolling\_mean\_30** | | --- | --- | --- | --- | | **0** | 1999-01-04 | 1.1789 | NaN | | **1** | 1999-01-05 | 1.1790 | NaN | | **2** | 1999-01-06 | 1.1743 | NaN | | **3** | 1999-01-07 | 1.1632 | NaN | | **4** | 1999-01-08 | 1.1659 | NaN | | **...** | ... | ... | ... | | **5694** | 2021-01-04 | 1.2296 | 1.211170 | | **5695** | 2021-01-05 | 1.2271 | 1.212530 | | **5696** | 2021-01-06 | 1.2338 | 1.213987 | | **5697** | 2021-01-07 | 1.2276 | 1.215357 | | **5698** | 2021-01-08 | 1.2250 | 1.216557 | |
| bol\_2006\_2009 = (euro\_dollar.Date.dt.year >= 2006) & (euro\_dollar.Date.dt.year <= 2009)  financial\_crisis = euro\_dollar[bol\_2006\_2009].copy()  bol\_2007\_2008 = (euro\_dollar.Date.dt.year >= 2007) & (euro\_dollar.Date.dt.year <= 2008)  financial\_crisis\_7\_8 = financial\_crisis[bol\_2007\_2008].copy()  financial\_crisis\_7\_8 = financial\_crisis[bol\_2007\_2008].copy()  style.use("fivethirtyeight") |
| fix, ax = plt.subplots(figsize=(8,3))  ax.plot(financial\_crisis["Date"],financial\_crisis.Rolling\_mean\_30,linewidth=1)  ax.plot(financial\_crisis\_7\_8["Date"],financial\_crisis\_7\_8.Rolling\_mean\_30,linewidth=3,color="#b00b1e")  ax.set\_xticklabels([])  ax.set\_yticklabels([])  #print(ax.get\_xticks())  #print(ax.get\_yticks())  ax.axvspan(xmin =14000 ,xmax= 14150,color="grey",alpha=0.3,ymin=0.01)  ax.grid(visible =False)  x = 13100  ys = ["2006","2007","2008","2009","2010"]  for year in ys:  ax.text(x=x,y=1.13,s=year,alpha =0.5)  x += 365  rates = ["1.2","1.3","1.4","1.5"]  y = 1.19  for rate in rates:  ax.text(x=13000,y=y,s=rate,alpha =0.5,fontsize=11)  y += 0.1  plt.show() |

Class 13

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  fandago = pd.read\_csv("fandango.csv")  ratings = pd.read\_csv("ratings\_16\_17.csv") |
| previous = fandago.copy()  after = ratings.copy() |
| previous = previous[['FILM','Fandango\_Stars','Fandango\_Ratingvalue','Fandango\_votes', 'Fandango\_Difference']]  previous   |  | **FILM** | **Fandango\_Stars** | **Fandango\_Ratingvalue** | **Fandango\_votes** | **Fandango\_Difference** | | --- | --- | --- | --- | --- | --- | | **0** | Avengers: Age of Ultron (2015) | 5.0 | 4.5 | 14846 | 0.5 | | **1** | Cinderella (2015) | 5.0 | 4.5 | 12640 | 0.5 | | **2** | Ant-Man (2015) | 5.0 | 4.5 | 12055 | 0.5 | | **3** | Do You Believe? (2015) | 5.0 | 4.5 | 1793 | 0.5 | | **4** | Hot Tub Time Machine 2 (2015) | 3.5 | 3.0 | 1021 | 0.5 | | **...** | ... | ... | ... | ... | ... | | **141** | Mr. Holmes (2015) | 4.0 | 4.0 | 1348 | 0.0 | | **142** | '71 (2015) | 3.5 | 3.5 | 192 | 0.0 | | **143** | Two Days, One Night (2014) | 3.5 | 3.5 | 118 | 0.0 | | **144** | Gett: The Trial of Viviane Amsalem (2015) | 3.5 | 3.5 | 59 | 0.0 | | **145** | Kumiko, The Treasure Hunter (2015) | 3.5 | 3.5 | 41 | 0.0 |   146 rows × 5 columns |
| after = after[['movie', 'year','fandango']]  after   | **movie** | **year** | **fandango** | | --- | --- | --- | | **0** | 10 Cloverfield Lane | 2016 | 3.5 | | **1** | 13 Hours | 2016 | 4.5 | | **2** | A Cure for Wellness | 2016 | 3.0 | | **3** | A Dog's Purpose | 2017 | 4.5 | | **4** | A Hologram for the King | 2016 | 3.0 | | **...** | ... | ... | ... | | **209** | X-Men: Apocalypse | 2016 | 4.0 | | **210** | XX | 2017 | 3.0 | | **211** | xXx: Return of Xander Cage | 2017 | 4.0 | | **212** | Zoolander 2 | 2016 | 2.5 | | **213** | Zootopia | 2016 | 4.5 | |
| previous['FILM'].str[-5:-1]  print(" ")  after['movie'].str[-5:-1]  0 Lan  1 Hour  2 lnes  3 rpos  4 Kin  ...  209 lyps  210 X  211 Cag  212 der  213 topi  Name: movie, Length: 214, dtype: object |
| previous['FILM'].str.extract(r'\((\d+)\)')[0]  0 2015  1 2015  2 2015  3 2015  4 2015  ...  141 2015  142 2015  143 2014  144 2015  145 2015  Name: 0, Length: 146, dtype: object  previous\_2015 = previous[previous["Year"]=="2015" ]  previous\_2015["Year"].unique()  array(['2015'], dtype=object)  0 5.0  previous["Fandango\_Stars"]  0 5.0  1 5.0  2 5.0  3 5.0  4 3.5  ...  141 4.0  142 3.5  143 3.5  144 3.5  145 3.5  Name: Fandango\_Stars, Length: 146, dtype: float64 |
| after["Year"] = after['movie'].str.extract(r'\((\d+)\)')[0]  \*after\_2016 = after[after["Year"]=="2016"]  after["fandango"]  0 3.5  1 4.5  2 3.0  3 4.5  4 3.0  ...  209 4.0  210 3.0  211 4.0  212 2.5  213 4.5  Name: fandango, Length: 214, dtype: float64 |
| plt.hist(previous["Fandango\_Stars"],label = "previous")  plt.hist(after["fandango"],label = "after")  plt.legend()  plt.show() |
| karnald density etimation plot import matplotlib.style as style  import numpy as np  plt.style.use("fivethirtyeight")  previous.Fandango\_Stars.plot.kde(label="2015",legend = True)  after.fandango.plot.kde(label="2015",legend = True)  plt.xlim(0,5)  plt.show() |
| previous.Fandango\_Stars.value\_counts().sort\_index()  3.0 12  3.5 27  4.0 41  4.5 55  5.0 11  Name: Fandango\_Stars, dtype: int64 |
| after.fandango.value\_counts(normalize=True).sort\_index()\*100  2.5 2.803738  3.0 8.411215  3.5 23.364486  4.0 38.317757  4.5 26.635514  5.0 0.467290  Name: fandango, dtype: float64 |
| mean\_2015 = previous.Fandango\_Stars.mean()  mean\_2015  4.089041095890411 |
| median\_2015 = previous.Fandango\_Stars.median()  median\_2015  4.0 |
| mode\_2015 = previous.Fandango\_Stars.mode()[0]  mode\_2015  4.5 |
| mean\_2016 = after.fandango.mean()  median\_2016 = after.fandango.median()  mode\_2016 = after.fandango.mode()[0]  avg["2015"] = [mean\_2015,median\_2015,mode\_2015]  avg["2016"] = [mean\_2016,median\_2016,mode\_2016]  avg.index=["mean","medain","mode"]  avg   |  | **2015** | **2016** | | --- | --- | --- | | **mean** | 4.089041 | 3.89486 | | **medain** | 4.000000 | 4.00000 | | **mode** | 4.500000 | 4.00000 | |
| avg["2015"].plot.bar(label="2015",color="#b00b13",align="center",width =0.25)  avg["2016"].plot.bar(label="2016",align="edge",width =0.25,figsize=(8,5),rot =0)  plt.legend(loc="upper center",framealpha = 0)  plt.show() |

Class 14

|  |
| --- |
| wnba = pd.read\_csv("wnba.csv")  wnba  wnba["Experience"].value\_counts().sort\_index()  1 12  10 5  11 4  12 4  13 3  14 3  15 2  2 11  3 19  4 11  5 14  6 14  7 4  8 3  9 11  R 23  Name: Experience, dtype: int64  wnba["Experience"] = wnba["Experience"].str.replace("R","0").astype(int)  wnba["Experience"].value\_counts().sort\_index()  0 23  1 12  2 11  3 19  4 11  5 14  6 14  7 4  8 3  9 11  10 5  11 4  12 4  13 3  14 3  15 2  Name: Experience, dtype: int64 |
| def exp\_ord(exp):  if exp == 0:  return "Rookie"  elif exp in range (0,4):  return "Little Experience"  elif exp in [0,4]:  return "Experience"  elif exp in range (6,11):  return "very Experience"  else:  return "Veteran"  wnba["Experience"].apply(exp\_ord)  0 Little Experience  1 Veteran  2 Experience  3 very Experience  4 Rookie  ...  138 very Experience  139 very Experience  140 Little Experience  141 very Experience  142 Little Experience  Name: Experience, Length: 143, dtype: object |
| wnba["Exp\_ordinal"] = wnba["Experience"].apply(exp\_ord)  wnba["Exp\_ordinal"].value\_counts()  Little Experience 42  very Experience 37  Veteran 30  Rookie 23  Experience 11  Name: Exp\_ordinal, dtype: int64  wnba["Exp\_ordinal"].value\_counts().iloc[[-2,0,2,1,-1]]  Rookie 23  Little Experience 42  Veteran 30  very Experience 37  Experience 11  Name: Exp\_ordinal, dtype: int64  wnba["Exp\_ordinal"].value\_counts().iloc[[-2,0,2,1,-1]].plot.barh(title= "Number of player in WNBA by level of expeience")  plt.show()  #use style of 538 remove the non data ink    wnba["Exp\_ordinal"].value\_counts().iloc[[-2,0,2,1,-1]].plot.pie(autopct = "% .2f%%" , title= "Number of player in WNBA by level of expeience")  plt.show() |
| wnba["Exp\_ordinal"].value\_counts(normalize=True).iloc[[-2,0,2,1,-1]]\*100  Rookie 16.083916  Little Experience 29.370629  Veteran 20.979021  very Experience 25.874126  Experience 7.692308  Name: Exp\_ordinal, dtype: float64 |
| import seaborn as sns  sns.countplot(x="Exp\_ordinal",hue="Pos",data=wnba,  order = ["Rookie","Little Experience","Experience","very Experience","Veteran"],  hue\_order = ["G","G/F","C","F/C","F"])  plt.show() |
| wnba["age\_cat"] = wnba["Age"].apply(lambda age:"old" if age>=27 else "Young" )  wnba["min\_cat"] = wnba["MIN"].apply(lambda Min:"Avg or Above" if Min>=497 else "below Avg")  sns.set\_theme()  sns.countplot(x="age\_cat",hue="min\_cat",data=wnba)  plt.show() |
| wnba.groupby("age\_cat")["MIN"].plot.hist()  plt.show() |
| wnba[wnba["Age"]>=27]["MIN"].plot.hist(label="Old",histtype ="step")  wnba[wnba["Age"]<27]["MIN"].plot.hist(label="Young",histtype ="step")  plt.axvline(497,label="Average")  plt.legend(loc = "lower left")  plt.show() |
| #overlap  wnba[wnba["Age"]>=27]["MIN"].plot.kde(label="Old")  wnba[wnba["Age"]<27]["MIN"].plot.kde(label="Young")  plt.axvline(497,label="Average")  plt.legend(loc = "lower left")  plt.show() |
| wnba["Pos"].unique()  array(['F', 'G/F', 'G', 'C', 'F/C'], dtype=object) |
| for pos in position:  bol\_pos = wnba["Pos"] == pos  wnba[bol\_pos]["Height"].plot.kde(label = pos)  plt.legend()  plt.show() |
| sns.stripplot(x="Pos",y="Height",data=wnba,jitter=False)  plt.show()  sns.boxplot(x="Pos",y="Height",data = wnba)  plt.show() |
| sns.boxplot(x="Pos",y="Height",data = wnba)  plt.show()  sns.boxplot(x="Pos",y="Weight",data = wnba)  plt.show()  sns.boxplot(x="Pos",y="Weight",data = wnba)  plt.show() |
| sns.boxplot(x="Pos",y="Weight",data = wnba)  plt.show() |
| wnba[wnba["Pos"]=="F"]["Weight"].describe()  count 33.000000  mean 83.090909  std 6.458346  min 71.000000  25% 79.000000  50% 82.000000  75% 86.000000  max 104.000000  Name: Weight, dtype: float64 |

Class 15

|  |
| --- |
| url = "https://github.com/fivethirtyeight/data/blob/master/star-wars-survey/StarWars.csv"  import requests  responce = requests.get(url)  responce  <Response [200]> |
| with open("StarWar.csv","w") as f:  f.write(responce.text)  responce.json().keys()  dict\_keys(['payload', 'title'])  data = responce.json()["payload"]["blob"]["rawLines"]  data[0] |
| import pandas as pd  with open("StarWar.csv","w") as f:  f.writelines(data)  with open("star.csv","w") as f:  for row in data:  f.write(row+"\n")  star = pd.read\_csv("star.csv",encoding="windows-1251")  star |
| df = star[1:].copy()  cleaned = df.iloc[:,0:3] loc does include last valuesiloc doesnot include last values mapping = {"Yes":True,"No":False}  cleaned.iloc[:,2] = cleaned.iloc[:,2].map(mapping)  renamed = ["id","seen\_any","fan"]  rename\_dict = dict(zip(cleaned.columns,renamed))  cleaned.rename(rename\_dict, axis =1)#, inplace =True   |  | **id** | **seen\_any** | **fan** | | --- | --- | --- | --- | | **1** | 3.292880e+09 | True | True | | **2** | 3.292880e+09 | False | NaN | | **3** | 3.292765e+09 | True | False | | **4** | 3.292763e+09 | True | True | | **5** | 3.292731e+09 | True | True | | **...** | ... | ... | ... | | **1182** | 3.288389e+09 | True | True | | **1183** | 3.288379e+09 | True | True | | **1184** | 3.288375e+09 | False | NaN | | **1185** | 3.288373e+09 | True | True | | **1186** | 3.288373e+09 | True | False |   1186 rows × 3 columns  cleaned.columns = renamed |
| cleaned.iloc[:,1:].value\_counts()  seen\_any fan  True True 552  False 284  dtype: int64  cleaned.iloc[:,1:].apply(pd.value\_counts,axis= "index")   |  | **seen\_any** | **fan** | | --- | --- | --- | | **True** | 936 | 552 | | **False** | 250 | 284 | |
| seen = df.iloc[:,3:9].copy()  seen.columns = ["Seen\_Eposode1","Seen\_Eposode2",  "Seen\_Eposode3","Seen\_Eposode4",  "Seen\_Eposode5","Seen\_Eposode6"]  seen = seen.notnull()  seen |
| ranking = df.iloc[:,9:15].copy()  ranking  ranking.columns = ["ranking\_"+str(i) for i in range(1,7)]  ranking.info()  ranking.astype(float) |
| final = cleaned.merge(seen,left\_index=True,right\_index=True)  final = final.merge(ranking,left\_index=True,right\_index=True)  final |
| cleaned.iloc[:,1:].apply(lambda col:col.value\_counts(normalize=True),axis= "index")  cleaned.iloc[:,1:]["seen\_any"].value\_counts(normalize=True)\*100 |
| def perc(col):  return col.value\_counts(normalize =True)  final.iloc[:,1:3].apply(perc,axis = 0)   |  | **seen\_any** | **fan** | | | --- | --- | --- | --- | | **True** | 0.789207 | 0.660287 | | **False** | 0.210793 | 0.339713 | |
| final.iloc[:,:9]  percentage = final.iloc[:,1:9].apply(lambda col:col.value\_counts(normalize=True),axis= "index")  import matplotlib.pyplot as plt  percentage.plot.bar()  plt.show()    percentage.loc[True].plot.barh()  plt.show() |
| final.iloc[:,9:] = final.iloc[:,9:].astype(float)  final.iloc[:,9:].mean()  ranking\_1 3.732934  ranking\_2 4.087321  ranking\_3 4.341317  ranking\_4 3.272727  ranking\_5 2.513158  ranking\_6 3.047847  dtype: float64  final.iloc[:,9:].mean().plot.barh()  plt.show() |
| male\_bol = df["Gender"] =="Male"  female\_bol = df["Gender"] =="Female"  male = final[male\_bol]  female = final[female\_bol]  male.shape ,female.shape  ((497, 15), (549, 15))  print(male.shape[0]+female.shape[0])  1046  df["Gender"].value\_counts(dropna =False)  Female 549  Male 497  NaN 140  Name: Gender, dtype: int64  male.iloc[:,9:].mean().plot.barh()  plt.show()  female.iloc[:,9:].mean().plot.barh()  plt.show()    female.iloc[:,9:].mean().plot.barh()  plt.show() |