De video waar ik mij op het gebaseerd om het project aan te vatten is: *Introduction to data visualization with Python.* Een cursus voorzien op PluralSight.

Om aan data visualisatie te doen wordt aangeraden om het *Anaconda* pakket te downloaden via https://www.anaconda.com/distribution/

In dit pakket zitten alle benodigdheden om met Python aan te slag te gaan om data visueel voor te stellen. Libraries die in dit pakket voorzien zijn:

* Matplotlib: wordt gebruikt om de data effectief te visualiseren
* Pandas: om data te importeren en te verwerken
* Jupyter: een interactieve Python notebook

Ik heb er voor gekozen om de 64-bit grafische interface voor Windows te installeren.

Eens geïnstalleerd kan je via *Anaconda Navigator,* ***Jupyter*** *Notebook* openen. Hiermee kan je code schrijven, organiseren en testen.

Om **Matplotlib** te gebruiken, dien je eerst de pyplot module te importeren. In de Jupyter Notebook geef je volgend commando in: *“from matplotlib import pyplot as plt”* Dit commando geeft aan dat je de pyplot module moet geïmporteerd worden vanuit matplotlib en plt moet genoemd worden.

Om vervolgens een eerste grafiek te tonen, kan je in Jupyter het volgende commando ingeven:

plt.plot([1,2,3],[1,4,9])

Wil je meerdere lijnen op een grafiek tonen:

plt.plot([1,2,3],[1,4,9])

plt.plot([1,2,3],[10,20,30])

Dit zegt dat je op de X-as punten 1, 2 ,3 moet gebruiken en op de Y-as punten 1, 4, 9. Om dit dan te tonen:

plt.show()

Labels toevoegen doe je op de volgende manier:

plt.xlabel('x-as')

plt.ylabel('y-as')

plt.title('test')

Een legend toevoegen:

plt.legend([‘set1’,’set2’])

Exporteren van de grafiek (afbeelding):

* Download afbeelding vanuit de browser
* Met het commando *plt.savefig(‘naamVanDeFile’)* het bestand wordt dan opgeslagen in dezelfde map als deze waar het project in zit.

**Pandas**

Eerst opnieuw imports doen:

from matplotlib import pyplot as plt

import pandas as pd

Eerst gaan we dummy data ingeven:

data = {'Year':[2008, 2012, 2016],

'attendees':[112,321,729],

'average age':[24,43,31]}

Vervolgens steken we deze data in een variabele om er een dataframe van te maken:

df = pd.DataFrame(data)

Wanneer je deze data dan toont met *df* krijg je de data set te zien in tabelvorm.

Om slecht één enkele kolom te selecteren, kan je dan df[‘Year’] (of data.Year) ingeven. Het resultaat is dan een *Panda series* datatype en is vergelijkbaar met een list in Python.

Hier kan je dan vergelijkingsoperatoren op toepassen: df['Year'] < 2013 die dan een reeks van Booleans zullen terug geven.

Wanneer je deze vergelijking in een nieuwe variabele steekt, dan kan je met df[‘nieuweVariabele’] de data van deze vergelijking laten zien. Wat Boolean Indexing wordt genoemd.

Om de waardes dan tov elkaar af te zetten en te tonen in een grafiek, kunnen we ons beroepen op het voorgaande:

plt.plot(df['Year'], df['attendees'])

plt.plot(df['Year'], df['average age'])

plt.legend(['attendees', 'average age'])

plt.show()

1. Gewicht sorteren

gewicht\_gesorteerd = sorted(set(data.gewicht))

1. Grafiek weergeven met de correlatie tussen lengte en gewicht

for x in gewicht\_gesorteerd:

y = data[data.gewicht == x]

plt.scatter(y.gewicht, y.lengte)

plt.xlim(19,31)

plt.ylim(110,140)

plt.xlabel('gewicht')

plt.ylabel('lengte')

plt.show()

1. **Hoe werd de dataset ingevoerd in Python?**

Om dit bestand te importeren in Python, hebben we gebruik gemaakt van twee verschillende manieren:

*Importeren vanuit een .csv bestand:*

De stappen die hiervoor werden uitgevoerd waren de volgende:

1. Het *Excel* bestand opslaan als een .csv bestand.
2. Openen van *Jupyter* en een nieuw project aanmaken. Om er voor te zorgen dat er later geen pad moet meegegeven worden naar het bestand, kan je best het nieuwe project opslaan in dezelfde map als de map waar het .csv bestand zich in bevindt.
3. Om een bestand te kunnen uitlezen, dient eerst *de Pandas library* geïmporteerd te worden.

import pandas as pd

1. De dataset inladen kan met het volgende commando:

pd.read\_csv(‘voetbal.csv’, sep=’;’)

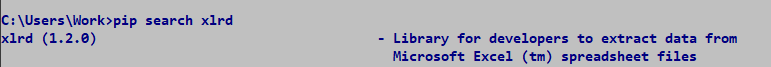
Het tweede argument, sep, dient om aan te geven welke separator wordt gebruikt in het .csv bestand.

1. Nu is de dataset wel ingeladen, maar om effectief bewerkingen toe te passen op deze dataset, dient het vorige commando aan een variabele toegekend te worden:

data = pd.read\_csv(‘voetbal.csv’, sep=’;’)

*Importeren vanuit een Excel bestand:*

Hiervoor werden volgende stappen uitgevoerd:

1. Installeer de xlrd library voor Python indien je deze niet hebt.  
     
     
   Installeren kan via de command line met commando: pip install xlrd
2. xlrd en *de Pandas library* importeren via:

import xlrd

import pandas as pd

1. Om de data in te lezen kies ik eerst het pad waar ik het databestand opgeslagen heb. Ik ga naar het pad en stel de current working directory in. Ik lees de data in met de functie read\_excel()

path='/Users/Work/Documents/TIN2/Data Advanced/PE/voetbal'

os.chdir(path)

cwd = os.getcwd()

data = pd.read\_excel("voetbal.xlsx", sheetname="gegevens")

(Wanneer het Excel bestand in dezelfde map zit, dient ook hier het pad niet meegegeven te worden.)

1. **Genereer zelf de geboortedatum (willekeurige datum gelegen tussen 01/01/2011 en 31/12/2011)**
2. **Genereer de kolom inzet met 3 mogelijke categorieën**

In de code die hieronder staat beschreven, worden de zowel de geboortedatum als de inzet gegenereerd. Dit is namelijk efficiënter dan voor de 2 stappen, aparte code te schrijven.

1. *Pandas* importeren en de dataset (= data) inladen.
2. Om wiskundige berekeningen te maken, dient ook de *numpy library* geïmporteerd te worden:

import numpy as np

1. Aangezien er ook met datums gaat gewerkt worden, gaat de *datetime library* ook geïmporteerd moeten worden:

import datetime as dt

1. Instellen van de start- en einddatum:

start = pd.to\_datetime('2011-01-01')

end = pd.to\_datetime('2011-12-31')

In dit geval wordt er aan start en end een *timestamp* toegekend. (De timestamp is een getal, weergegeven in nanoseconden sinds 1 januari 1970).

start\_u = pd.to\_datetime('2011-01-01')

end = pd.to\_datetime('2011-12-31')

1. Een *timestamp* kan gebruikt worden om random datums te genereren omdat de *value* van een *timestamp* weergegeven wordt in nanoseconden. De twee *values* kunnen dan gebruikt worden als ranges om random getallen tussen te berekenen. Getallen die vervolgens weer kunnen omgezet worden naar datums.

start\_u = start.value//10\*\*9

end\_u = end.value//10\*\*9

Om de gegenereerde getallen nadien terug om te zetten naar datums gaat er gebruik gemaakt worden van seconden. Dit de reden waarom hier de *value* gedeeld wordt door 10^9.

1. Omdat de kolom inzet, strings gaat bevatten, gaan we dit zo aangeven aan python:

data[‘inzet’] = ‘’

1. Als laatste stap zal met een iteratie, beide kolommen gevuld worden. Om het formaat van de datum op een leesbare manier weer te geven, zal gebruik gemaakt worden van de functie .strftime()

for x, row in data.iterrows():

y = (pd.to\_datetime(np.random.randint(start\_u, end\_u), unit='s'))

data.at[x,'geboortedatum'] = dt.date(y.year, y.month, y.day).strftime('%d/%m/%Y')

if y < pd.to\_datetime('2011-04-01'):

data.at[x, 'inzet'] = 'zeer goed'

elif y < pd.to\_datetime('2011-10-01'):

data.at[x, 'inzet'] = 'goed'

else:

data.at[x, 'inzet'] = 'matig'

Dit geeft uiteindelijk volgend resultaat (enkel de eerste 5 resultaten worden getoond):



1. **Maak in Python de grafiek die op tabblad met naam ‘grafiek’ ziet**

In eerste instantie moeten er een aantal imports gedaan worden.

* import pandas as pd
* import mathplotlib.pyplot as plt
  + Deze import wordt gedaan om de grafiek te kunnen weergeven in python.
* import numpy as np

De grafiek die weergegeven wordt, geeft de correlatie weer tussen lengte en gewicht. Om er voor te zorgen dat er buiten deze waardes, nog extra gegevens kunnen afgeleid worden, werden ook de posities van de spelers en het aantal goals dat iedere speler heeft gemaakt, opgenomen in de data om de grafiek te genereren.

De volgende code werd gebruikt:

naam = data["naam"]

positie = data["positie"].unique()

col = {'staart':'red', 'linkervleugel':'blue', 'rechtervleugel':'green','piloot':'black', 'keeper':'yellow'}

colors = data['positie'].apply(lambda x: col[x])

goals = data["aantal gemaakte goalen"]

geboortedatum = data["geboortedatum"]

inzet = data['inzet']

gewicht = data['gewicht']

lengte = data["lengte"]

aantalSpelers = len(naam)

plt.scatter(gewicht, lengte, s=(goals\*200)+100, alpha=0.4, c=colors)

##plt.legend(('staart','linkervleugel', 'rechtervleugel', 'piloot', 'keeper'), ('red', 'blue', 'green', 'black', 'yellow'), loc='upper left')

##s=goals voor punten groter te maken naargelang aantal goals

##c=positie geeft iederen positie een kleur

##plt.bar(goals, positie, width=0.8)

## voor staafdiagrammen

for i in range(aantalSpelers):

if goals[i] >= 3:

plt.text(gewicht[i],lengte[i]+0.1,naam[i][6:]).set\_fontsize(10)

##label iedere dot met zijn positie, +0.1 om iets boven dot uit te komen

##fontsize aanpassen aan goals

plt.xlabel("Gewicht")

plt.ylabel("Lengte")

l1 = plt.scatter(20, 135, alpha=0.6, c = 'red')

l2 = plt.scatter(20, 135, alpha=0.6, c = 'blue')

l3 = plt.scatter(20, 135, alpha=0.6, c = 'green')

l4 = plt.scatter(20, 135, alpha=0.6, c = 'black')

l5 = plt.scatter(20, 135, alpha=0.6, c = 'yellow')

plt.legend((l1, l2, l3, l4, l5), ('Staart', 'Linkervleugel', 'Rechtervleugel', 'Piloot', 'Keeper'), loc='upper left')

plt.show()

Met volgende grafiek als resultaat:

Afbeelding met schermafbeelding

Automatisch gegenereerde beschrijving

1. **Maak in python een staafdiagram van het aantal gemaakte goalen per positie. Deel de posities telkens ook nog op naar geboortecategorie en bespreek**

# Split up values by birth category

cat1= data[pd.to\_datetime(data.geboortedatum) < pd.to\_datetime('2011-04-01')]

cat2= data[(pd.to\_datetime('2011-04-01')<=pd.to\_datetime(data.geboortedatum)) & (pd.to\_datetime(data.geboortedatum)< pd.to\_datetime('2011-07-01'))]

cat3= data[(pd.to\_datetime('2011-07-01')<=pd.to\_datetime(data.geboortedatum)) & (pd.to\_datetime(data.geboortedatum) < pd.to\_datetime('2011-10-01'))]

cat4= data[pd.to\_datetime(data.geboortedatum) >= pd.to\_datetime('2011-10-01')]

# retreiving positions and initializing bars

bars1 = {}

bars2 = {}

bars3 = {}

bars4 = {}

positions = []

for position in data['positie']:

if position not in positions:

positions.append(position)

bars1[position]=0

bars2[position]=0

bars3[position]=0

bars4[position]=0

# Values of each position in all birthcategories

for x, row in cat1.iterrows():

position=cat1.at[x, 'positie']

bars1[position]=bars1[position]+ cat1.at[x, 'aantal gemaakte goalen']

for x, row in cat2.iterrows():

position=cat2.at[x, 'positie']

bars2[position]=bars2[position]+ cat2.at[x, 'aantal gemaakte goalen']

for x, row in cat3.iterrows():

position=cat3.at[x, 'positie']

bars3[position]=bars3[position]+ cat3.at[x, 'aantal gemaakte goalen']

for x, row in cat4.iterrows():

position=cat4.at[x, 'positie']

bars4[position]=bars4[position]+ cat4.at[x, 'aantal gemaakte goalen']

# Heights of bars1 + bars2 and bars1 + bars2 + bars3

bars1\_2 = np.add(list(bars1.values()), list(bars2.values())).tolist()

bars1\_2\_3= np.add(bars1\_2,list(bars3.values())).tolist()

# The position of the bars on the x-axis

r = range(0,len(positions))

# Names of group and bar width

names = positions

barWidth = 1

# Create green bars

plt.bar(r, list(bars1.values()), color='#2BCE48', edgecolor='white', width=barWidth)

# Create orange bars (middle no.1), on top of the first ones

plt.bar(r, list(bars2.values()), bottom=list(bars1.values()), color='#FF8800', edgecolor='white', width=barWidth)

# Create blue bars (midlle no.2)

plt.bar(r, list(bars3.values()), bottom=bars1\_2, color='#0075DC', edgecolor='white', width=barWidth)

# Create red bars (top)

plt.bar(r, list(bars4.values()), bottom=bars1\_2\_3, color='#FF0010', edgecolor='white', width=barWidth)

# legend, ticks and labels

plt.legend(["januari-maart","april-juni","juli-september","oktober-december"])

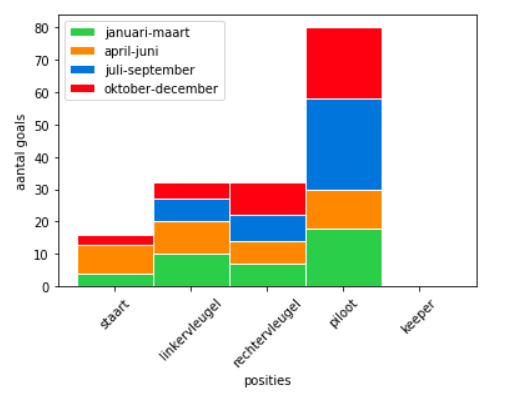
plt.xticks(range(0,len(positions)),positions, rotation=45)

plt.ylabel("aantal goals")

plt.xlabel("posities")

# Show graphic

plt.show()



1. **Bereken mbv Python het gemiddelde en de modus van kolom D (aantal gemaakte goalen) per positie**
2. **Bereken mbv Python kwartiel 1 en de standaardafwijking van kolom G (het gewicht)**

weights=list(data['gewicht'])

weights.sort()

q1=weights[len(weights)//4]

print("kwartiel 1 van de gewichten is "+str(q1))

print()

import statistics

standard\_deviation= statistics.stdev(weights)

print("de standaardafwijking van de gewichten is " + str(standard\_deviation))

Resultaat:

kwartiel 1 van de gewichten is 22.1

de standaardafwijking van de gewichten is 2.8835752219434228