# PORTFOLIO ASSIGNMENT 101 & 102

Tristan Goossens (2188507)
Avans Hogeschool

# Contents

# 1.0 Wat is data science

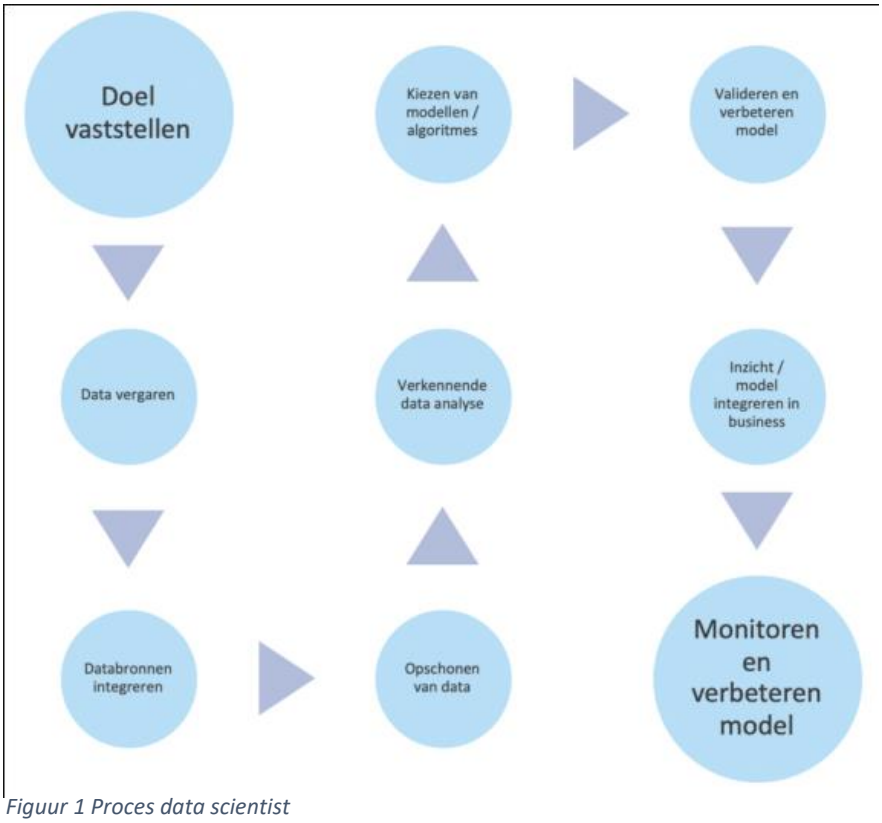
Data science is een interdisciplinair vakgebied dat wetenschappelijke methoden, processen, algoritmen en systemen combineert om inzichten en kennis uit gegevens te halen. Het combineert elementen van statistiek, informatica en domeinspecifieke kennis om complexe datasets te analyseren, interpreteren en communiceren.

In het algemeen omvat data science de volgende stappen:

- Gegevensverzameling: Verzamelen van relevante gegevens uit verschillende bronnen, zoals databases, API's en sensoren.
- Gegevensvoorbereiding: Het reinigen, transformeren en voorbereiden van de gegevens voor analyse.
- Verkennende data-analyse: Het onderzoeken en visualiseren van de gegevens om patronen, correlaties en andere inzichten te identificeren.
- Modelbouw: Het creëren van statistische of machine learning-modellen om de gegevens te analyseren en voorspellingen te doen.
- Model evaluatie: Het beoordelen van de prestaties van de modellen en het verfijnen ervan indien nodig.
- Implementatie: Het implementeren van de modellen in real-world omgevingen en het gebruiken ervan om besluitvorming te informeren.

Data science wordt veel gebruikt in industrieën zoals financiën, gezondheidszorg, marketing en productie, evenals in onderzoeksgebieden zoals sociale wetenschappen en natuurwetenschappen.

In de komende hoofdstukken zal het proces die een data scientist doorloopt om zijn doel te bereiken en wat data science inhoud beschreven worden aan de hand van een aantal voorbeelden uit het portfolio.




*Figuur 1 Proces data scientist*

## 2.0 Doel

Het doel van de analyse is om meer inzicht te krijgen over een dataset naar mijn keuze. Het doel dat ik gesteld heb is dat ik de verschillen wil zien in de data van een spel op laag niveau en een pot op erg hoog niveau.

## 3.0 Data vergaren

De dataset die ik voor mijn portfolio heb gebruikt heb ik binnengehaald via Kaggle. Kaggle is een website waar duizenden datasets te vinden zijn die gemaakt zijn door gebruikers van de website. (Chess dataset). De gekozen dataset is via Kaggle als .csv bestand gedownload

### 3.1 Voorbeeld portfolio

In figuur 2 is te zien hoe de data in een Jupyter notebook is geladen en is enkele informatie over de kolommen getoond

**4.1 Chess dataset**

This is a set of just over 20,000 games collected from a selection of users on the site Lichess.org

```
In [1]: import pandas as pd

        chess = pd.read_csv("../assets/chess.csv", sep=',')
```

```
In [2]: chess.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20058 entries, 0 to 20057
Data columns (total 16 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              20058 non-null  object
 1   rated           20058 non-null  bool
 2   created_at      20058 non-null  float64
 3   last_move_at    20058 non-null  float64
 4   turns           20058 non-null  int64
 5   victory_status  20058 non-null  object
 6   winner          20058 non-null  object
 7   increment_code  20058 non-null  object
 8   white_id        20058 non-null  object
 9   white_rating    20058 non-null  int64
 10  black_id        20058 non-null  object
 11  black_rating    20058 non-null  int64
 12  moves           20058 non-null  object
 13  opening_eco     20058 non-null  object
 14  opening_name    20058 non-null  object
 15  opening_ply     20058 non-null  int64
dtypes: bool(1), float64(2), int64(4), object(9)
memory usage: 2.3+ MB
```

*Figuur 2 Vergaren data*

## 4.0 Databronnen integreren

De databron is zoals in kop 2.0 beschreven gedownload via Kaggle. Ook is in hoofdstuk 2 te zien hoe de databron al in een Jupyter notebook ingeladen kon worden

## 5.0 Opschonen van data

Het opschonen van data was in deze opdracht niet van toepassing. Ook had ik het geluk dat de data consistent was en er waren nergens lege kolommen te vinden. In figuur 2 is te zien hoe elke kolom evenveel 'Non-null' waarden had

## 6.0 Verkennende data analyse

Als onderzoeker weet je nog niet helemaal welke resultaten je gaat vinden en welke kant je onderzoek opgaat. Je gebruikt deze analyse om beter bekend te raken met je dataset.

De verschillende tools binnen een verkennende analyse zijn bijvoorbeeld:

- Scatter plot
- Bar plot
- Box plot
- Pie charts

### 6.2 Voorbeelden portfolio

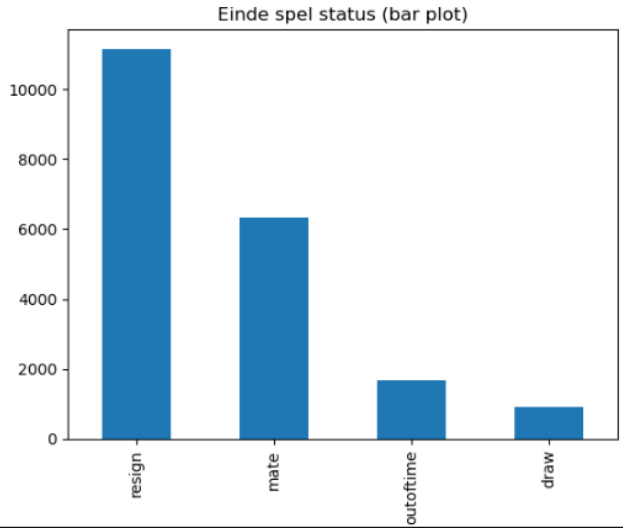
**6.2.1 Victory status**

Gegeven die aangeeft hoe het spel is beeindigd

```
In [2]: chess['victory_status'].unique()
Out[2]: array(['outoftime', 'resign', 'mate', 'draw'], dtype=object)

In [3]: chess['victory_status'].value_counts()
Out[3]: resign       11147
        mate          6325
        outoftime     1680
        draw           906
        Name: victory_status, dtype: int64

In [4]: chess['victory_status'].value_counts().plot(kind='bar', title='Einde spel status (bar plot)')
Out[4]: <AxesSubplot:title={'center':'Einde spel status (bar plot)'}>
```




*Figuur 1 Univariate analyse categorische data*

### 6.3 Univariate analysis (Numerical data)

- Turns (Aantal zetten tijdens het potje)
- combined_rating (Rating van wit en zwart samen)

#### 6.3.1 combined_rating

De rating van beide spelers (wit en zwart) bij elkaar opgeteld. Deze kolom bestaat niet in de dataset. Deze moet dus eerst aangemaakt worden met de som (Rating zwart + Rating wit)

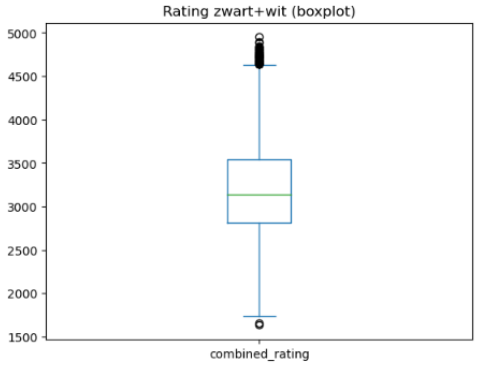
```
In [14]: chess['combined_rating'] = chess['white_rating'] + chess['black_rating']

In [15]: chess['combined_rating'].describe()

Out[15]: count    20058.000000
         mean      3185.463855
         std        526.347651
         min       1633.000000
         25%       2813.000000
         50%       3137.500000
         75%       3542.000000
         max       4951.000000
         Name: combined_rating, dtype: float64

In [16]: chess['combined_rating'].plot(kind='box', title='Rating zwart+wit (boxplot)')

Out[16]: <AxesSubplot:title={'center':'Rating zwart+wit (boxplot)'}>
```




*Figuur 2 Univariate analyse m.b.v een boxplot*

#### 10.1.1 Correlaties

De kolommen waar ik voor gekozen heb is combined_rating (rating_white + rating_black) en opening_ply (het aantal theoretische zetten van zwart en wit). Deze hebben namelijk de sterkste positieve correlatie.

```
In [2]: chess[['combined_rating', 'opening_ply', 'turns']].corr().style.background_gradient(cmap='coolwarm', axis=None).format(precision
```

Out[2]:

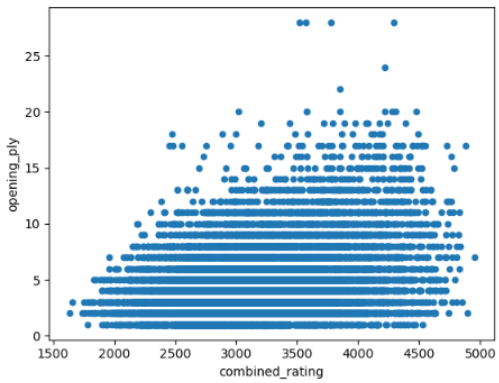
| | combined_rating | opening_ply | turns |
|---|---|---|---|
| combined_rating | 1.00 | 0.29 | 0.16 |
| opening_ply | 0.29 | 1.00 | 0.06 |
| turns | 0.16 | 0.06 | 1.00 |

#### 10.1.2 Scatterplot correlatie

De sterkste positieve correlatie is vervolgens gevisualiseerd in een scatterplot. Hier in is te zien dat spelers met een lagere gecomineerde rating vaak beginnen met minder zetten uit theorie. Naar mate hoe hoger de gecombineerde rating is er te zien hoe er meer zetten vanuit theorie worden gespeeld. Dit is ook het verwachtte resultaat.

```
In [3]: chess.plot(kind='scatter', x='combined_rating', y='opening_ply')

Out[3]: <AxesSubplot:xlabel='combined_rating', ylabel='opening_ply'>
```

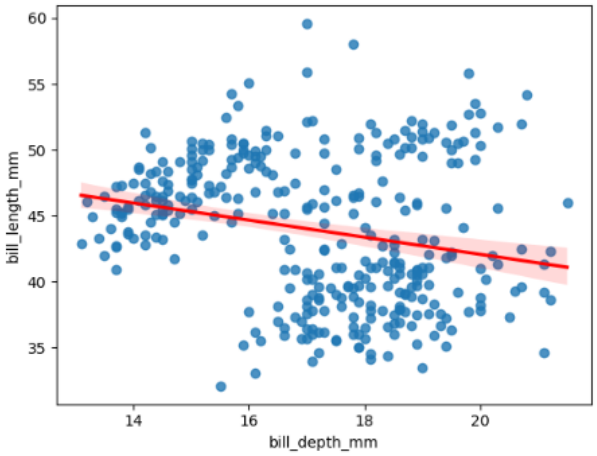



*Figuur 3 Bivariate analyse m.b.v een scatterplot*

### 9.3 Weakest correlation

De zwakste correlatie ligt tussen de kolommen bill_depth_mm en bill_length_mm

```
In [5]: sns.regplot(data=penguins, x='bill_depth_mm', y='bill_length_mm', fit_reg=True, line_kws={"color": "red"})
Out[5]: <AxesSubplot:xlabel='bill_depth_mm', ylabel='bill_length_mm'>
```




*Figuur 4 Correlatie in een scatterplot*

# 7.0 Model trainen, valideren en verbeteren

Een data scientist kan een model trainen op basis van bestaande gegevens om zo nieuwe meetwaarden te categoriseren. Het model maakt dan een ‘gok’ op basis van nauwkeurigheid

## 7.2 Voorbeeld portfolio

**16.2 Train model**

In [3]:
```
from sklearn.tree import DecisionTreeClassifier

features= ['white_rating', 'black_rating','turns']
dt_classification = DecisionTreeClassifier(max_depth=10) # Increase max_depth to see effect in the plot
dt_classification.fit(chessTrainSet[features], chessTrainSet['winner'])
```

Out[3]:
```
DecisionTreeClassifier(max_depth=10)
```

**16.3 Evaluate trained model**

In [4]:
```
def calculate_accuracy(predictions, actuals):
    if(len(predictions) != len(actuals)):
        raise Exception("The amount of predictions did not equal the amount of actuals")

    return (predictions == actuals).sum() / len(actuals)
```

In [5]:
```
predictionsOnTrainset = dt_classification.predict(chessTrainSet[features])
predictionsOnTestset = dt_classification.predict(chessTestSet[features])

accuracyTrain = calculate_accuracy(predictionsOnTrainset, chessTrainSet['winner'])
accuracyTest = calculate_accuracy(predictionsOnTestset, chessTestSet['winner'])

print("Accuracy on training set " + str(accuracyTrain))
print("Accuracy on test set " + str(accuracyTest))
```

```
Accuracy on training set 0.6852564102564103
Accuracy on test set 0.6010302426055167
```

*Figuur 5 getrainde data model met evaluatie*