educX

Leitfaden für nachvollziehbare Schritte

1. Kurze Darstellung des Problembereichs / Aufriss des Themas

1.1 Inhaltlich

In dieser Arbeit wird der Datensatz High School Alcoholism and Academic Performance untersucht. Er beinhaltet 31 Variablen zum Umfeld, Verhalten und Wünschen der befragten Schüler zweier portugiesischer Schulen.

Mit Hilfe dieses Datensatzes und guten Darstellungen in Diagrammen und Zusammenfassungen, soll ein Einblick darüber ermöglicht werden, welchen Einfluss Alkoholismus auf die schulischen Leistungen hat und ob es eventuell wichtigere Einflüsse aus dem Umfeld der Schüler auf deren Leistung gibt.

1.2 Begründung des Themas

Darstellung der Relevanz des Themas?

Durch den Erfolg von maschineller Produktion und steigender Automatisierung verändert sich der Jobmarkt weltweit und geht weg von körperlicher hin zu geistiger Arbeit. Um die Schüler bestmöglich auf diese Zukunft vorzubereiten, soll sichergestellt werden, dass jedem Schüler möglichst gute Rahmenbedingungen für gute akademische Leistungen zur Verfügung gestellt werden. Um heraus zu finden, welche Stellräder für dieses Ziel die größte Relevanz haben, sollen gute Umfragen und Auswertungen dieser durchgeführt werden. Mit den relevantesten Einflüssen, die man verändern kann, kann dann ein Plan zur Verbesserung des schulischen Umfeldes erdacht werden.

Darstellung eines persönlichen Erkenntnisinteresses.

Da eine kontinuierliche Weiterbildung im Leben des modernen Arbeiters heutzutage unabdingbar ist, ist es nützlich, heraus zu finden, in welchem Umfeld die besten Bedingungen zum erreichen guter akademischer Leistungen gegeben sind und welchen Einfluss Alkoholkonsum auf diese hat und ob es bedeutendere Einflüsse als Alkohol gibt.

2. Nachvollziehbare Schritte

2.1 Der Stand der Forschung / Auswertung der vorhandenen Literatur / Tutorials ...

Wie man hier sieht, wurde das Thema auch schon von anderen Nutzern untersucht.

Außerdem gibt es zum Beispiel für die Suchtprävention ähnliche Umfragen und Untersuchungen, die Auswertungen durchgeführt haben. Jedoch dürfte bei diesen der Fokus nicht auf den schulische Leistungen liegen, sondern darauf, dass der Schulalltag die Schüler nicht in die Sucht treibt.
So wurde in anderen Umfragen der Fokus auf Schulunlust, Stress und gefühltes Schulversagen in Zusammenhang mit Alkoholkonsum untersucht, der Einfluss des Umfeldes des Schülers scheint dabei nicht so vordergründig gewesen zu sein.

Alkoholkonsum scheint meist unter dem Aspekt der Realitäts-Flucht/Stressbewältigung verstanden zu werden. Wohingegen es aber auch Zusammenhänge wie: „Gute schulische Leistung → Gutes Gefühl/Man fühlt sich nicht schuldig mehr tun zu müssen → man macht mehr mit Freunden → man trinkt mehr“ geben könnte. Wodurch das Umfeld dann eher zu Alkoholkonsum führen könnte, als die schulischen Leistungen, je nach dem, ob Schule oder Freunde wichtiger sind.

2.2 Fragestellung

1. Können Daten einer Umfrage so visualisiert werden, dass man auf einen Blick ein Verständnis für die Daten entwickeln kann?
2. Kann man in den Daten einer Umfrage Zusammenhänge zwischen dem Umfeld der Schüler, ihren Noten und Alkoholkonsum erkennen?

2.3 Methode

2.3.1 Daten vorbereiten

```
1  ##########################################
   Zelle Ausführen | Unten ausführen | Zelle debuggen
2  # %% Load data into pandas data-frame
3
4  from collections import Counter
5  import seaborn as sns
6  import matplotlib.pyplot as plt
7  import pandas as pd
8
9  # en_lpor_explorer.csv - High School Alcohol and Academic Performance
10 filename: str = './data/en_lpor_explorer.csv'
11
12 # try except block in case the file does not exist
13 try:
14     # Read csv data into pandas dataframe
15     data_frame = pd.read_csv(filename)
16 except:
17     print(f'File cannot be opened: {filename}')
18
19
20 # en_lpor_classification.csv - High School Alcohol and Academic Performance
21 filename_classification: str = './data/en_lpor_classification.csv'
22
23 # try except block in case the file does not exist
24 try:
25     # Read csv data into pandas dataframe
26     data_frame_classification = pd.read_csv(filename_classification)
27 except:
28     print(f'File cannot be opened: {filename_classification}')
29
30
```

- Alle benötigten Bibliotheken importieren
- Dateinamen in Variable speichern

- In einem try-except Block versuchen die CSV-Datei in einen Pandas Dataframe zu laden
  - Falls die Datei nicht existiert den Anwender darauf hinweisen
- Wiederholen für weitere CSV-Datei, die die Daten als eindeutige Nummer klassifiziert enthält

Head/Kopf der Datensätze ausgeben mit:

```
30
31  print(data_frame.head(1))
32  print(data_frame_classification.head(1))
```

Ergibt:

```
3                School  Gender  Age Housing_Type Family_Size Parental_Status  \
4   0  Gabriel Pereira  Female   18        Urban     Above 3       Separated
5
6      Mother_Education  Father_Education Mother_Work Father_Work  ... Is_Dating  \
7   0  Higher Education  Higher Education   Homemaker     Teacher  ...        No
8
9     Good_Family_Relationship Free_Time_After_School Time_with_Friends  \
10  0                     Good              Moderate              High
11
12    Alcohol_Weekdays Alcohol_Weekends Health_Status School_Absence  \
13  0         Very Low         Very Low          Fair              4
14
15    Grade_1st_Semester Grade_2nd_Semester
16  0                  0                11
17
18  [1 rows x 31 columns]
19     School  Gender  Age  Housing_Type  Family_Size  Parental_Status  \
20  0       0       1   18             0            1                1
21
22     Mother_Education  Father_Education  Mother_Work  Father_Work  ...  \
23  0                 4                 4            3            0  ...
24
25     Is_Dating  Good_Family_Relationship  Free_Time_After_School  \
26  0          0                         4                       3
27
28     Time_with_Friends  Alcohol_Weekdays  Alcohol_Weekends  Health_Status  \
29  0                  4                 1                 1              3
30
31     School_Absence  Grade_1st_Semester  Grade_2nd_Semester
32  0               4                   0                  11
33
34  [1 rows x 31 columns]
```

2.3.2 Histogram plotting Funktion vorbereiten

```
34  ########################################
    Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [142]
35  # %% Histogram function
36
37
38  def show_distribution(data, title: str):
39      # Sort data ascending
40      data_sorted = data.sort_values(ascending=True)
41
42      # Show histogram
43      sns.displot(data_sorted).set(title=title)
44
45      # Show counts of identical items
46      counts_sorted = Counter(data_sorted)
47
48      counts_string = [
49          f'{counted[0]}: {counted[1]} participants - {round(100 * counted[1] / counts_sorted.total(), 2)}% of total
              participants' for counted in counts_sorted.items()]
50
51      print('\n'.join(counts_string))
52
53
```

- Definiere ‘show_distribution()’ Funktion mit Parametern ‘data’ und ‘title’
- Innerhalb der Funktion werden
  - Die Daten aufsteigend sortiert
  - Via Seaborn ‘displot()’ als Histogram geplottet
  - Bei dem durch ‘set(title)’ der Titel des Diagrams auf den gegebenen Parameter gesetzt wird
  - Dann werden durch Counter() die Anzahl der einzigartigen Symbole in den Daten gezählt
  - Diese Zahlen werden dann für jedes Symbol angezeigt, sowie der Anteil den das Symbol an der Gesamtmenge hat → siehe weiter unten, für eine Beispiel Ausgabe dieser Funktion

2.3.3 Schul und Geschlechts Histogramm plotten

```
54  ########################################
    Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [143]
55  # %% Show surveyed schools
56
57  # Extract school data
58  data_school = data_frame['School']
59
60  # Display it
61  title = 'Surveyed Schools and participation numbers'
62  print(f'\n{title}:\n')
63  show_distribution(data_school, title=title)
64
65
66  ########################################
    Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [144]
67  # %% Show surveyed sexes
68
69  # Extract school data
70  data_sexes = data_frame['Gender']
71
72  # Display it
73  title = 'Gender of surveyed participants'
74  print(f'\n{title}:\n')
75  show_distribution(data_sexes, title=title)
76
```

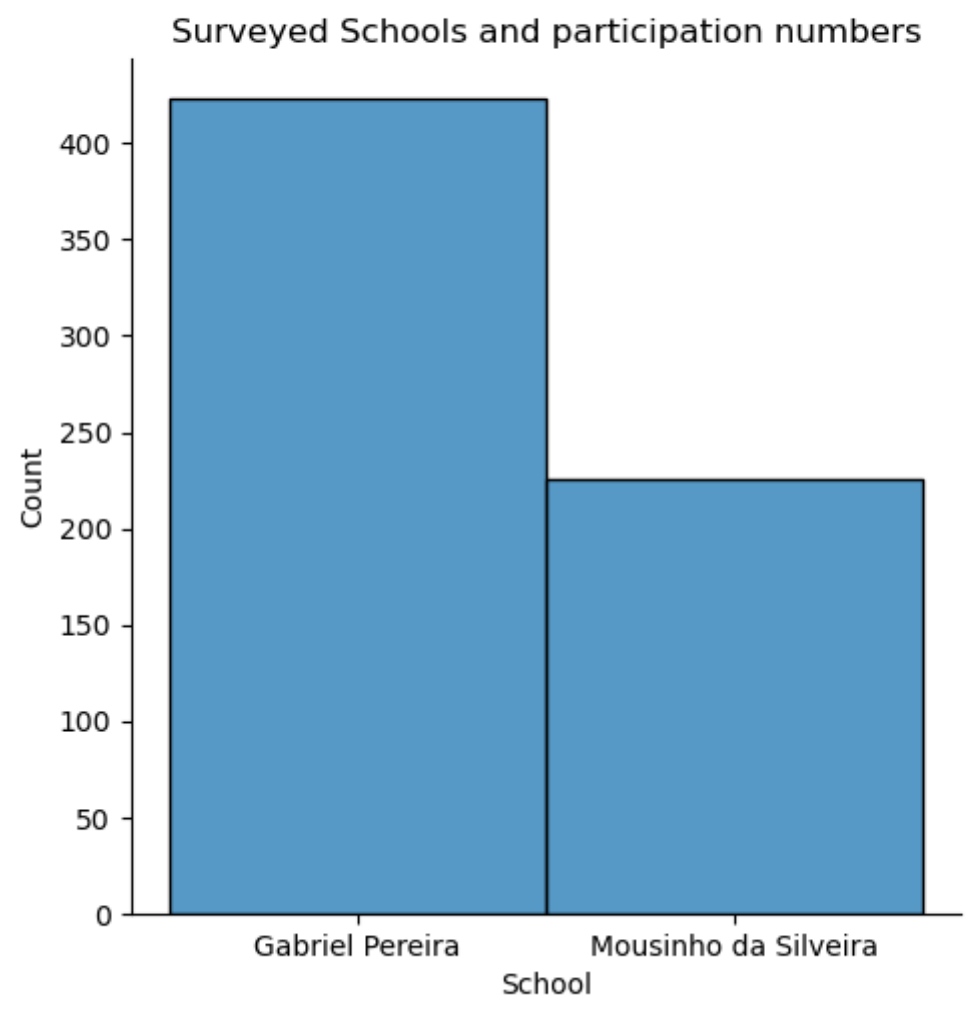
- Extrahieren der Spalten ‘School’ und ‘Gender’ in separate Variablen

- Setzten eines Diagramm-Titels auf eine Variable sowie printen dieses Titels für bessere Übersichtlichkeit
- Aufrufen der Histogramm Plott-Funktion mit den extrahierten Daten und dem Titel

Plotten ergibt (Hier sieht man auch den Text, den die ‘show_distribution’ Funktion aus den Counts erzeugt):

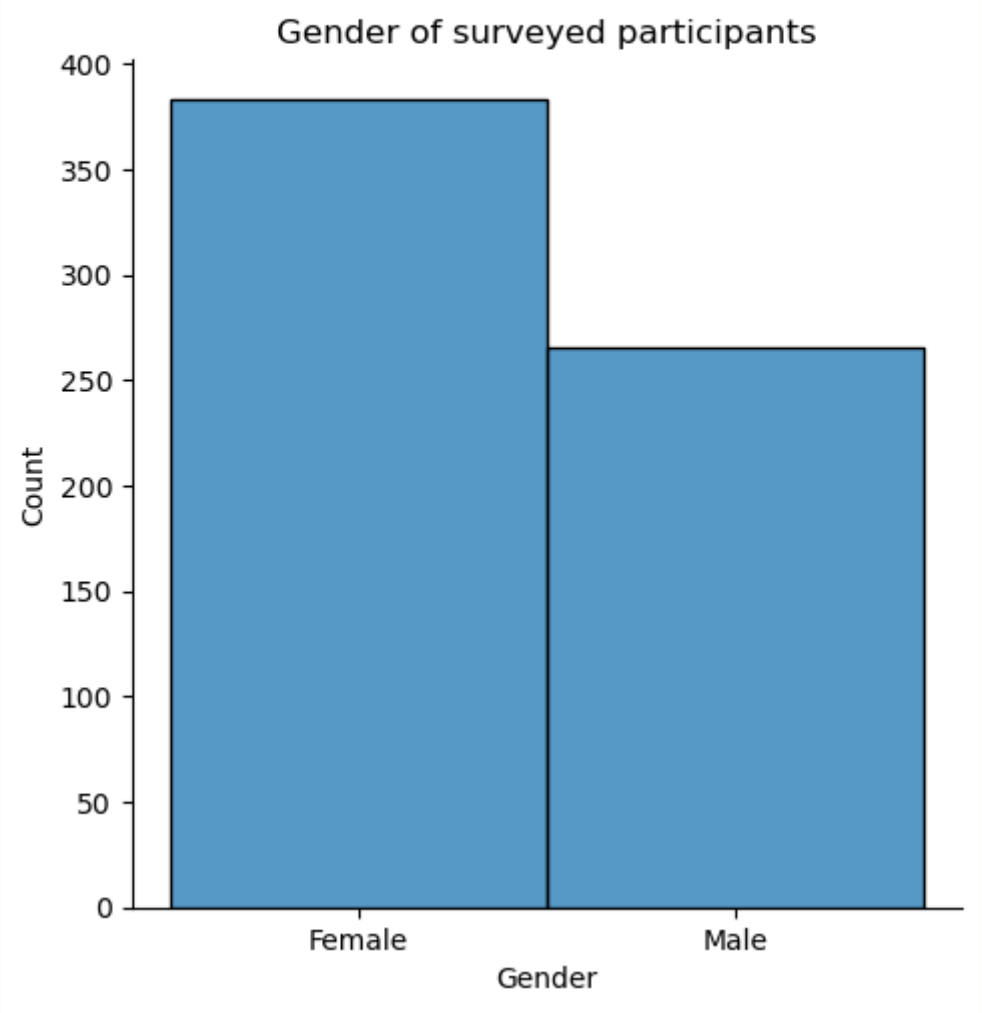
```
Surveyed Schools and participation numbers:

Gabriel Pereira: 423 participants - 65.18% of total participants
Mousinho da Silveira: 226 participants - 34.82% of total participants
```



```
Gender of surveyed participants:

Female: 383 participants - 59.01% of total participants
Male: 266 participants - 40.99% of total participants
```



2.3.4 Altersverteilung plotten

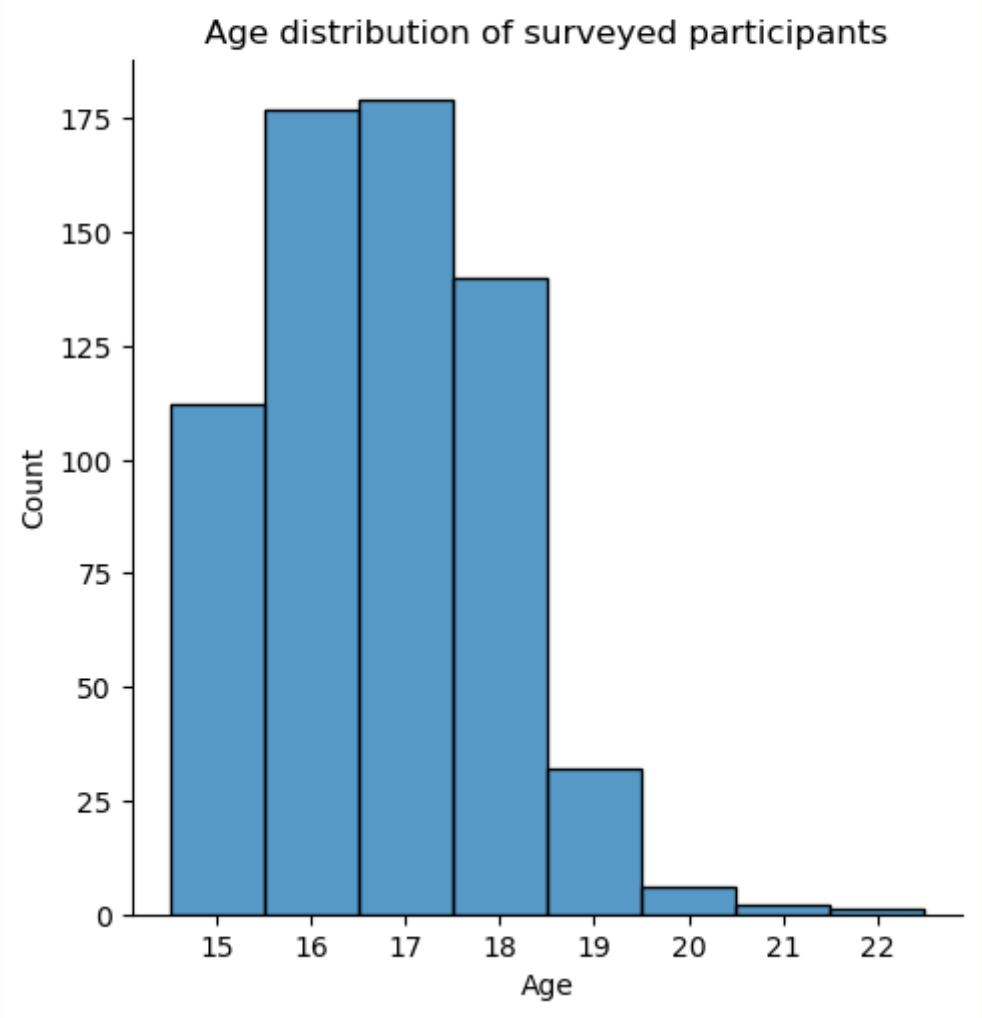
```
78  ########################################
    Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [145]
79  # %% Show surveyed ages
80
81  # Extract school data
82  data_ages = data_frame['Age']
83
84  # Cast to string so that seaborne does not try to use the actual number values as x axis but rather interprets those
85  # values as unique symbols thus center aligning bar and tick
86  data_ages_strings = data_ages.astype(str)
87
88  # Display it
89  title = 'Age distribution of surveyed participants'
90  print(f'\n{title}:\n')
91  show_distribution(data_ages_strings, title=title)
92
```

- Bei der Altersverteilung gibt es eine Besonderheit
- Nach der Extraktion der Alters-Spalte aus dem Dataframe muss diese noch in String-Werte konvertiert/gecasted werden
- Tut man dies nicht, interpretiert Seaborn die Werte als Integers oder Floats und will damit die Balken genau an dieser Stelle auf der X-Achse zeichnen, wodurch das Histogramm sehr merkwürdig aussieht
- Nach der Konvertierung interpretiert Seaborn die Werte als eindeutige Symbole und stellt die Balken und Ticks wie gewünscht zentriert zueinander dar

Histogramm:

```
Age distribution of surveyed participants:

15: 112 participants - 17.26% of total participants
16: 177 participants - 27.27% of total participants
17: 179 participants - 27.58% of total participants
18: 140 participants - 21.57% of total participants
19: 32 participants - 4.93% of total participants
20: 6 participants - 0.92% of total participants
21: 2 participants - 0.31% of total participants
22: 1 participants - 0.15% of total participants
```



2.3.5 Automatische Generation der Histrogramme aller restlichen Spalten

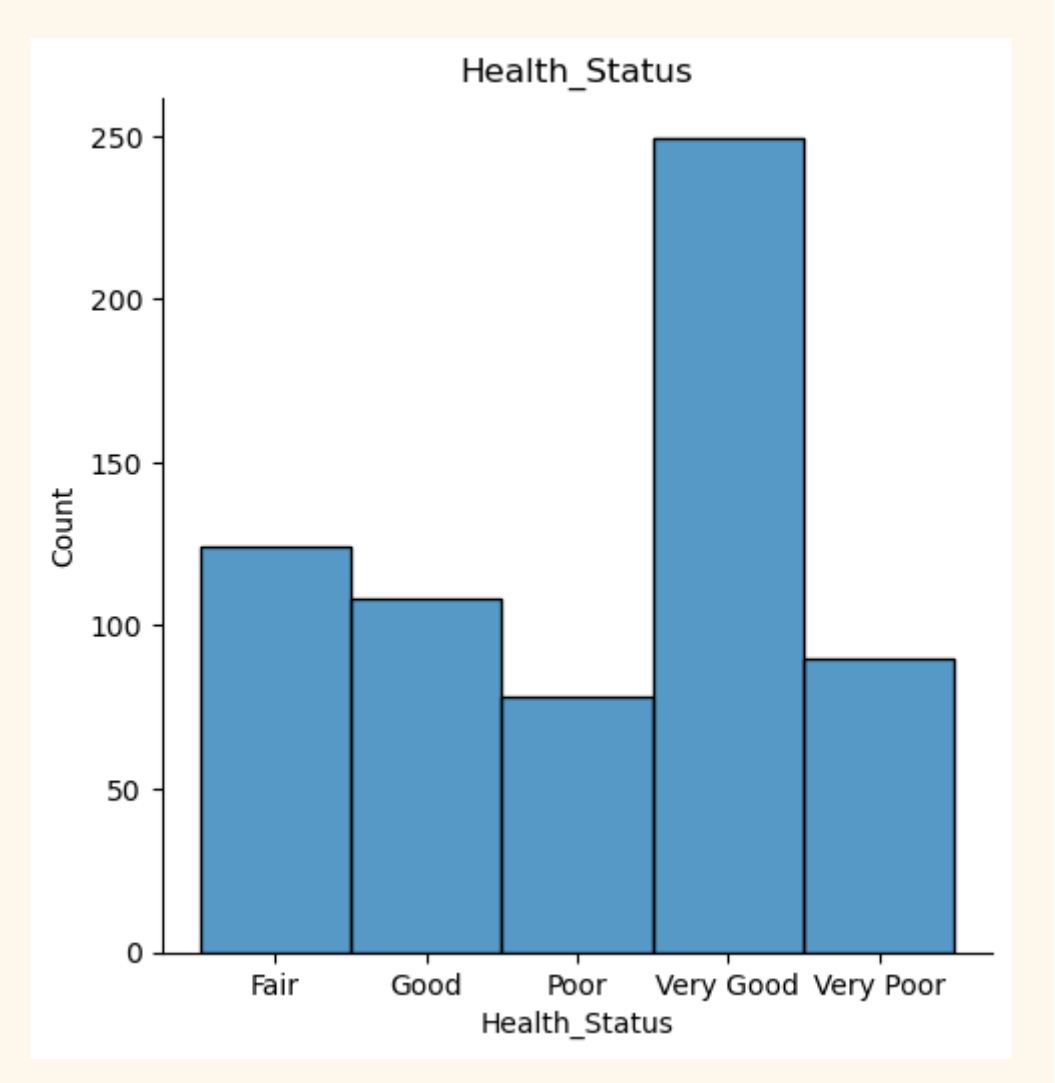
- Lade alle Spalten des Dataframes in eine Variable

- Definiere Array ‘without’ welches die Strings der schon manuell erstellten Diagramme enthält

```
94  ########################################
    Zelle Ausführen | Oben laufen | Zelle debuggen
95  # %% Generate all other histograms 'automatically'
96  columns = data_frame.columns
97  without = ['School', 'Gender', 'Age']
98
99  filtered_columns = columns.difference(without)
100
101 for column in filtered_columns:
102     title = column
103     print(f'\n{title}:\n')
104
105     # casting all data to strings to prevent misaligned bars and ticks
106     data_casted = data_frame[column].astype(str)
107
108     show_distribution(data=data_casted, title=title)
109
110 # unfortunately the auto generated diagrams aren't as readable as some of the values aren't ordered quite right and some
111 # labels overlap - however for a quick overview of the data it is still useful
112
```

- Filtere diese durch anwenden der ‘difference()’ Funktion aus den Spalten heraus
- Gehe dann via For-Schleife durch alle Spalten-Namen hindurch und
  - Setzte die Variable Title auf den aktuellen Spalten-Namen und gebe diesen aus
  - Zur Sicherheit werden alle Spalten-Werte aus oben beschriebenen Gründen zu String-Werten gecasted/konvertiert
  - Dann wird die ‘show_distribution()’ Funktion mit den Vorbereiteten Variablen aufgerufen

Erzeugt zum Beispiel:



- Man sieht, dass man damit grundsätzlich schon ein gutes Verständniss der Daten gewinnen kann, jedoch ist hier die Reihenfolge der Symbole auf der X-Achse doch etwas ungewöhnlich
- Dies könnte man verbessern, wenn man das Diagramm “von Hand” etwas verbessert und nicht auf die generelle Funktion zurück greift
- Das war für die Auswertung jedoch nicht wichtig genug, deswegen reichen hier die ‘automatisch’ generierten Diagramme aus

2.3.6 Erkentnissgewinn durch Korrelation

```
114  ########################################
     Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [147]
115  # %% Find correlated values
116  correlation = data_frame_classification.corr()
117
118  plt.figure(figsize=(12, 8))
119  # "PiYG" is a good divergent color map (white in the middle and pink and green at the extremes)
120  # in combination with the vmin and vmax declaration this helps to put the white color in the middle at 0, pink at -1
121  # (negative correlating) and green at +1 positive correlation
122  sns.heatmap(correlation, cmap='PiYG', vmin=-1,
123              vmax=1).set(title='Correlation Matrix of all Survey Data')
124
125  # show unique correlations as table but without the 'diagonal'/'self-correlation'
126  correlation_table = correlation[correlation < 1].unstack().transpose().sort_values(
127      ascending=False).drop_duplicates()
128  pd.options.display.max_rows = 5000
129  print(correlation_table)
130
131  # show correlations of data with alcohol consumption
132  print(
133      f'\n\nCorrelations with alcohol consumption on weekdays:\n{correlation["Alcohol_Weekdays"][correlation
         ["Alcohol_Weekdays"] < 1].sort_values(ascending=False).drop_duplicates()}')
134  print(
135      f'\n\nCorrelations with alcohol consumption on weekends:\n{correlation["Alcohol_Weekends"][correlation
         ["Alcohol_Weekends"] < 1].sort_values(ascending=False).drop_duplicates()}')
136
137  # show correlations of data with grades
138  print(
139      f'\n\nCorrelations with 1st semester grades:\n{correlation["Grade_1st_Semester"][correlation["Grade_1st_Semester"]
         < 1].sort_values(ascending=False).drop_duplicates()}')
140  print(
141      f'\n\nCorrelations with 2nd semester grades:\n{correlation["Grade_2nd_Semester"][correlation["Grade_2nd_Semester"]
         < 1].sort_values(ascending=False).drop_duplicates()}')
142
```

- Achtung, damit die Korrelation gut funktioniert werden hier die bereits Klassifizierten Daten verwendet
- Auf diese wird die ‘corr()’ Funktion angewendet und das Ergebniss in einer Variable gespeichert
- Dann wird dem Plotter gesagt, dass er eine größere Figur zeichnen soll
- Und via Seaborn wird eine heatmap aus den Korrelations-Daten erzeugt
  - Damit man im resultierenden Diagramm die Beziehungen besser erkennen kann wird hier ein divergenter Farbverlauf ‘PiYG’ (möglicherweise Pink Yellow Green) ausgewählt, sowie die min und max Werte angegeben, damit die neutrale Farbe Yellow bei 0 in der Mitte liegt – da wo keine Korrelation zwischen den Werten besteht → dadurch kann man jeglichen grün- oder pink-Stich in der jeweiligen Zelle sofort als positive oder negative Korrelation erkennen (siehe folgendem Bild)
- Dann werden die Korrelationen noch gruppiert und gefiltert auf eindeutige Werte ohne die Selbstkorrelations-Diagonale als Tabelle ausgegeben
- Da dies eine sehr lange Tabelle ist und man sich möglicherweise auch für geringe Korrelationen interessiert setzt der Code den Wert, ab dem Seaborn die Ausgabe abschneidet noch etwas höher
- Und zum Schluss werden noch ein paar besonders interessante Korrelationen einzeln ausgegeben, nämlich alle Korrelationen zum
  - Alkoholkonsum unter der Woche
  - Alkoholkonsum am Wocheende

- Noten erstes Semester
- Noten zweites Semester

Dies erzeugt:

```
Correlations with alcohol consumption on weekdays:
Alcohol_Weekends                0.616561
Time_with_Friends               0.245126
School_Absence                  0.172952
Age                             0.134768
Legal_Responsibility            0.117029
Free_Time_After_School          0.109904
Commute_Time                    0.092824
Is_Dating                       0.062042
Health_Status                   0.059067
Private_Tutoring                0.051986
Housing_Type                    0.047304
School                          0.047169
Has_Internet                    0.042811
Reason_School_Choice            0.026597
Extracurricular_Activities      0.022592
Father_Education                0.000061
Mother_Education               -0.007018
Mother_Work                    -0.011831
Parental_Educational_Support   -0.016844
Extra_Educational_Support      -0.028076
Father_Work                    -0.039808
Parental_Status                -0.041513
Family_Size                    -0.060482
Good_Family_Relationship       -0.075767
Attended_Daycare               -0.078376
Desire_Graduate_Education      -0.131663
Weekly_Study_Time              -0.137585
Grade_2nd_Semester             -0.189480
Grade_1st_Semester             -0.195171
Gender                         -0.282696
Name: Alcohol_Weekdays, dtype: float64
```

2.3.7 Definition Bell-Curve Funktion

```python
##########################################
Zelle Ausführen | Oben laufen | Zelle debuggen
# %% bell curve function
def show_bell_curve(data, title: str | None = None, xlabel: str | None = None, axvline: int | None = None):
    # Sort data ascending
    data_sorted = data.sort_values(ascending=True)

    # Show histogram
    gfg = sns.displot(data_sorted, kind='kde')

    # set labels if given, use default if not
    if title:
        gfg.set(title=title)
    if xlabel:
        gfg.set(xlabel=xlabel)

    if axvline:
        # plot vertical line if given
        plt.axvline(x=axvline)

        # and display amount of values above and below it
        below_axvline = [v for v in data_sorted if v < axvline]
        at_above_axvline = [v for v in data_sorted if v ⩾ axvline]
        print(
            f'Count below line {len(below_axvline)} - {round(100 * len(below_axvline) / len(data_sorted), 2)}%')
        print(
            f'Count at and above line {len(at_above_axvline)} - {round(100 * len(at_above_axvline) / len(data_sorted),
                2)}%', '\n')

    # Show counts of identical items
    counts_sorted = Counter(data_sorted)

    counts_string = [
        f'{counted[0]}: {counted[1]} participants - {round(100 * counted[1] / counts_sorted.total(), 2)}% of total
            participants' for counted in counts_sorted.items()]

    print('\n'.join(counts_string))
```

- Definiere ‘show_bell_curve’ Funktion mit dem Parameter ‘data’, sowie den optionalen Parametern ‘title’, ‘xlabel’ und ‘axvline’ welche durch ‘= None’ optional sind
- Sortiere die Daten
- Plotte mit Seaborn ein displot, jedoch diesmal mit “kind=’kde’”, also einem anderen Diagramm-Typ
- Da die Funktion optionale Parameter entgegen nimmt wird hier jeweils mit “if” geprüft, ob diese Werte enthalten und falls ja, werden diese entsprechend verarbeitet
  - Setzten des Titels
  - Setzten der X-Achsen-Beschriftung
  - Zeichnen einer vertikalen Linie
- Falls eine vertikale Linie gezeichnet wird werden außerdem die Anzahl aller Werte aus den Daten, die kleiner als die Linie sind gezählt und deren Prozentanteil ausgegeben, sowie für die Werte, die auf der Linie oder größer sind
- Dann wird wie bei den Histogrammen gezählt und ausgegeben, wie häufig jeder eindeutige Wert vorkam und dessen Prozentanteil

Beispiel siehe Verwendung für die Noten-Verteilung, die folgt.

2.3.8 Darstellung Notenverteilung via Bell-Curve Funktion

```
180  ########################################
     Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [149]
181  # %% Show bell curve of 1st semester grades
182
183  # Extract school data
184  data_grades_1st = data_frame['Grade_1st_Semester']
185  # Cast to int to make sure the values are numerical
186  data_grades_1st_int = data_grades_1st.astype(int)
187
188  # Display it
189  # (Under the Portuguese system, grades are given on a scale from 0 to 20, the minimum passing grade being 10.)
190  title = 'Normal distribution of 1st semester grades'
191  print(f'\n{title}:\n')
192  show_bell_curve(data_grades_1st_int, title=title,
193                  xlabel='Grades in 1st semester (minimum passing grade: 10)', axvline=10)
194
195
196  ########################################
     Zelle Ausführen | Oben laufen | Zelle debuggen | Gehe zu [150]
197  # %% Show bell curve of 2nd semester grades
198
199  # Extract school data
200  data_grades_2nd = data_frame['Grade_2nd_Semester']
201  # Cast to int to make sure the values are numerical
202  data_grades_2nd_int = data_grades_2nd.astype(int)
203
204  # Display it
205  # (Under the Portuguese system, grades are given on a scale from 0 to 20, the minimum passing grade being 10.)
206  title = 'Normal distribution of 2nd semester grades'
207  print(f'\n{title}:\n')
208  show_bell_curve(data_grades_2nd_int, title=title,
209                  xlabel='Grades in 2nd semester (minimum passing grade: 10)', axvline=10)
210
```

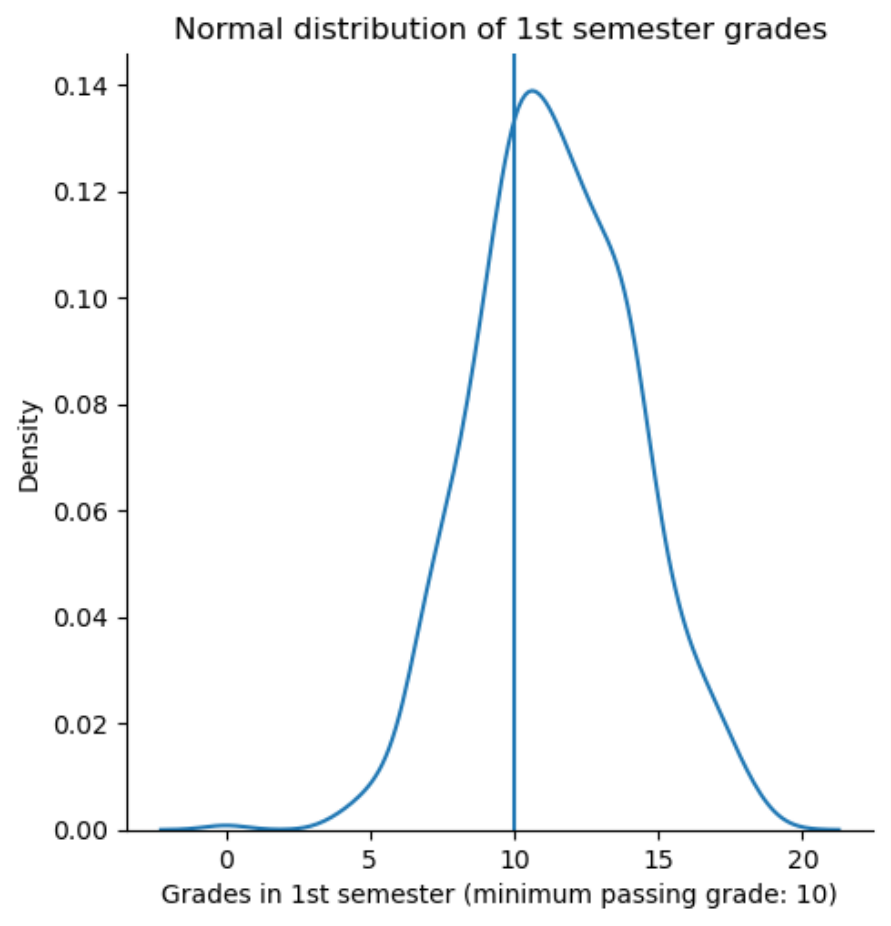
- Extraktion der erst und zweit Semester Noten
- Konvertierung zu integern zur Sicherheit falls diese anders geladen wurden
- Setzen und aus gegen des Titels der Diagramme
- Aufrufen der ‘show_bell_curve’ Funktion mit den jeweiligen Noten-Daten-Sätzen, Titeln und Labeln, sowie dem Wert, an dem die vertikale Linie gezeichnet werden soll (hier zur Verdeutlichung, wie viele Noten über der Bestehensgrenze lagen)

Erzeugt für das erste Semester:

```
Normal distribution of 1st semester grades:

Count below line 157 - 24.19%
Count at and above line 492 - 75.81%

0: 1 participants - 0.15% of total participants
4: 2 participants - 0.31% of total participants
5: 5 participants - 0.77% of total participants
6: 9 participants - 1.39% of total participants
7: 33 participants - 5.08% of total participants
8: 42 participants - 6.47% of total participants
9: 65 participants - 10.02% of total participants
10: 95 participants - 14.64% of total participants
11: 91 participants - 14.02% of total participants
12: 82 participants - 12.63% of total participants
13: 72 participants - 11.09% of total participants
14: 71 participants - 10.94% of total participants
15: 35 participants - 5.39% of total participants
16: 22 participants - 3.39% of total participants
17: 16 participants - 2.47% of total participants
18: 7 participants - 1.08% of total participants
19: 1 participants - 0.15% of total participants
```



2.4 Ergebnisse

Die Ergebnisse der einzelnen Code-Abschnitte wurden jeweils nach der Beschreibung der Code Blöcke eingefügt.
Zusammenfassen kann man sagen, dass die textuelle/tabellarische Darstellung der Korrelationen die größten Einflüsse auf die Schulische Leistung am offensichtlichsten zeigt.
Aus diesen Tabellen konnten folgende Einflüsse als am relevantesten herausgefunden werden.

1. Positive Korrelation mit den Noten:
    1. Desire Graduate Education
    2. Weekly Study Time
    3. Mother Education
    4. Father Education

2. Negative Korrelation mit den Noten:
    1. School
    2. Alcohol Weekdays
    3. Mother Work
    4. Age
    5. Housing Type
    6. Commute Time

In Worten ausgedrückt bedeutet das, dass der Wunsch eines Hochschulabschlusses der beste Indikator für gute Noten ist, danach kommt die wöchentliche Lernzeit sowie der Bildungsstand der Mutter und des Vaters.
Schlecht für gute Noten hingegen sind die Schule auf die man geht, trinken von Alkohol (unter der Woche), der Job der Mutter, das Alter, sowie die Wohnsituation und die Schulweglänge.

2.5 Ausblick

Die hier verwendeten Methoden können auch für ähnliche Datensätze angewendet werden. Außerdem kann man noch weiter gehen und zum Beispiel versuchen den Datensatz noch weiter zu gruppieren. Beispielsweise nach kombinationen von Einflüssen, welche die besten Ergebnisse erziehlen, damit man dann Aussagen treffen könnte wie: “Der Wunsch eines Hochschulabschlusses ist wichtiger als ein kurzer Schulweg und eine gute Wohnsituation zusammen.” Oder: “Geringerer Alkoholkonsum bringt weniger als gut gebildete Eltern”.
Das könnte man zeigen, indem man die Daten jeweils in Gruppen mit hohem und geringem Hochschulabschluss Wunsch aufteilt und dann deren Noten plotted, sowie Gruppen mit kurzem Schulweg und guter Wohnsituation, bezieungsweise langem Schulweg und schlechter Wohnsituation und dann zeigt, wie groß die Unterschiede in den Noten sind.
Eventuell kann man daraus dann Leitsätze ableiten um auch weniger ‘glücklichen’ Schülern Hoffnung zu geben. Zum Beispiel durch die Aussage: “Selbst wenn Ihr aktuell in ärmeren Verhältnissen aufwachst, kann allein der Wunsch nach einer guten Lebenssituation als Erwachsener viel mehr bringen, als wenn Ihr jezt

schon in besseren Verhältnissen leben würdet.