# Projekt: Employment

1. Vorüberlegungen:
   - Bei der Auswahl des Datensatzes habe ich mich für den Bereich des Employments entschieden. Es sollen hier Klassifizierungen angewendet werden.
   - Employment: Unternehmen haben sich in Ihrer Struktur in den vergangenen Jahren stark verändert. Arbeitnehmer, aber auch Arbeitgeber handeln beim Thema Kündigungen kurzfristiger als in der Vergangeheit. Die Gründe dafür sind vielfältig. Folglich wird es in Zukunft für ein Unternehmen wichtiger sein, besser beurteilen zu können, wann ein Mitarbeiter aller Wahrscheinlichkeit nach kündigen wird, um möglichst effizient die passenden Mitarbeiter zu beschäftigen.
   - In diesem Projekt soll es also darum gehen, nach einem bestehenden Datensatz ein Machine Learning Modell zu erstellen, welches möglichst genau durch Klassifizierung vorhersagen kann, ob ein Mitarbeiter kündigen wird oder nicht.
2. Ziel:
   - ✔ Mit einer Auswahl von verschiedenen Algorithmen aus dem Machine-Learning-Bereich erstelle ich ein ML-Modell, mit welchem ich bestmöglich eine präzise Vorhersage treffen kann, ob ein Mitarbeiter aufgrund seiner individuellen Gegebenheiten im Unternehmen in naher Zukunft kündigen wird oder nicht.

## Inhaltsverzeichnis

# . Datensatzanalyse - Kündigung eines Mitarbeiters bei einem Unternehmen

## 1. Kostenfreien Datensatz bei Kaggle importieren

- Datensatz eines Unternehmens
- Herunterladen und Einlesen das Datensatzes im .csv-Format

## 2. Analyse des Datensatzes

A) Datensatz mit seinen spezifischen Voraussetzungen:

- 33 unterschiedliche Features (columns) mit 1470 Samples (rows)
- ein Feature „Attrition“, welches man als Target nutzen kann („Label“)
- Features Inhalt:
  - ✗ „Age“ = Alter des Mitarbeiters
  - ✗ „Attrition“ = gekündigt ja/nein
  - ✗ „BusinessTravel“ = geschäftlich unterwegs viel/wenig...
  - ✗ „Department“ = IAbteilung
  - ✗ „DistanceFromHome“ = Distanz des Wohnortes
  - ✗ „Gender“ = Geschlecht
  - ✗ „JobInvolvement“ = Beteiligungsbewertung
  - ✗ „JobLevel“ = Jobebene
  - ✗ „JobRole“ = Verantwortlichkeiten
  - ✗ „JobSatisfaction“ = Zufriedenheitsbewertung
  - ✗ „MaritalStatus“ = Familienstand
  - ✗ „MonthlyIncome“ = Monatliches Einkommen
  - ✗ „NumCompaniesWorked“ = Anzahl der Unternehmen
  - ✗ „OverTime“ = Überstunden ja/nein
  - ✗ „PercentSalaryHike“ = Gehaltserhöhungen
  - ✗ „PerformanceRating“ = Leistungsbewertung
  - ✗ „StockOptionLevel“ = Aktienbeteiligung
  - ✗ „TotalWorkingYears“ = Arbeitsjahre
  - ✗ „TrainingTimesLastYear“ = Schulungen

- ✗ „YearsAtCompany“ = Arbeitsjahre im Unternehmen
- ✗ „YearsSinceLastPromotion“ = letzten Beförderung
- ✗ „YearsWithCurrManager“ = Zeit unter dem aktuellen Manager
- ✗ „Higher_Education“ = Bildungsniveau
- ✗ „Date_of_Hire“ = Einstellungsdatum
- ✗ „Date_of_termination“ = Datum der Kündigung
- ✗ „Status_of_leaving“ = Grund für den Austritt
- ✗ „Mode_of_work“ = HomeOffice ja/nein
- ✗ „Leaves“ = Urlaubstage
- ✗ „Absenteeism“ = Abwesenheitstage
- ✗ „Work_accident“ = Arbeitsunfall ja/nein
- ✗ „Source_of_hire“ = Einstellungsquelle
- ✗ „Job_Mode“ = Vollzeit-/Teilzeit- oder Vertragsarbeit

B) den Datensatz im Detail untersuchen:

- notwendige Bibliotheken zur Bearbeitung des Datensatzes importieren (Numpy, Pandas, MatPlotLib, Seaborn...)
- Daten in Jupyter Notebook importieren und DataFrame (Pandas) erstellen mit pd.read_csv("...")
- Datensatzausschnitt:

| | Age | Attrition | BusinessTravel | Department | DistanceFromHome | Gender | JobInvolvement | JobLevel | JobRole | JobSatisfaction | ... | Date_of_Hire | Date_of |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 37 | Yes | Travel_Rarely | Research & Development | 2 | Male | 2 | 1 | Laboratory Technician | 3 | ... | 21-01-2021 | |
| 1 | 21 | No | Travel_Rarely | Research & Development | 15 | Male | 3 | 1 | Research Scientist | 4 | ... | 13-03-2021 | |
| 2 | 45 | No | Travel_Rarely | Research & Development | 6 | Male | 3 | 3 | Research Director | 1 | ... | 23-01-2021 | |
| 3 | 23 | No | Travel_Rarely | Sales | 2 | Male | 3 | 1 | Sales Representative | 1 | ... | 25-04-2021 | |
| 4 | 22 | No | Travel_Rarely | Research & Development | 15 | Female | 3 | 1 | Laboratory Technician | 4 | ... | 14-06-2021 | |

5 rows × 33 columns

- mit .columns die Feature-Names ermitteln und mit .info die Werte ermitteln → Nullwerte, kategorische/numerische Werte

## 3. Daten preparieren/bereinigen

A) Überblick verschaffen: → welche unterschiedliche Datentypen gibt es?

```
In [7]:  1 df02.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 33 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   Department                1470 non-null   object
 4   DistanceFromHome          1470 non-null   int64
 5   Gender                    1470 non-null   object
 6   JobInvolvement            1470 non-null   int64
 7   JobLevel                  1470 non-null   int64
 8   JobRole                   1470 non-null   object
 9   JobSatisfaction           1470 non-null   int64
 10  MaritalStatus             1470 non-null   object
 11  MonthlyIncome             1470 non-null   int64
 12  NumCompaniesWorked        1470 non-null   int64
 13  OverTime                  1470 non-null   object
 14  PercentSalaryHike         1470 non-null   int64
 15  PerformanceRating         1470 non-null   int64
 16  StockOptionLevel          1470 non-null   int64
 17  TotalWorkingYears         1470 non-null   int64
 18  TrainingTimesLastYear     1470 non-null   int64
 19  YearsAtCompany            1470 non-null   int64
 20  YearsSinceLastPromotion   1470 non-null   int64
 21  YearsWithCurrManager      1470 non-null   int64
 22  Higher_Education          1470 non-null   object
 23  Date_of_Hire              1470 non-null   object
 24  Date_of_termination       0 non-null      float64
 25  Status_of_leaving         1470 non-null   object
 26  Mode_of_work              1470 non-null   object
 27  Leaves                    1470 non-null   int64
 28  Absenteeism               1470 non-null   int64
 29  Work_accident             1470 non-null   object
 30  Source_of_Hire            1470 non-null   object
 31  Job_mode                  1470 non-null   object
 32  Unnamed: 32               0 non-null      float64
dtypes: float64(2), int64(17), object(14)
memory usage: 379.1+ KB
```

B) Wie viele unterschiedliche Werte sind in den einzelnen Features?

```
In [9]:  1 df02.nunique()

Out[9]: Age                         43
        Attrition                    2
        BusinessTravel               3
        Department                   3
        DistanceFromHome            29
        Gender                       2
        JobInvolvement               4
        JobLevel                     5
        JobRole                      9
        JobSatisfaction              4
        MaritalStatus                3
        MonthlyIncome             1349
        NumCompaniesWorked          10
        OverTime                     2
        PercentSalaryHike           15
        PerformanceRating            2
        StockOptionLevel             4
        TotalWorkingYears           40
        TrainingTimesLastYear        7
        YearsAtCompany              37
        YearsSinceLastPromotion     16
        YearsWithCurrManager        18
        Higher_Education             4
        Date_of_Hire              1112
        Date_of_termination          0
        Status_of_leaving            5
        Mode_of_work                 2
        Leaves                       6
        Absenteeism                  4
        Work_accident                2
        Source_of_Hire               4
        Job_mode                     3
        Unnamed: 32                  0
        dtype: int64
```

C) Wie lauten diese einzelnen Werte?

```
In [11]:  1  df02["BusinessTravel"].value_counts()
Out[11]: BusinessTravel
         Travel_Rarely        1043
         Travel_Frequently     277
         Non-Travel            150
         Name: count, dtype: int64
```

D) kategorische Features in numerische umwandeln mit df[„...“].replace({"...":0,"...":1},inplace=True)

E) NULL-Werte, falls existierend, entfernen oder wenn sinnvoll, durch Durchschnittswerte (mit Boxplot überprüfen!) über eine Funktion und .apply() ersetzen

F) nicht relevante Features entfernen

G) der Datensatz beinhaltet nach Bereinigung/Aufbereitung 28 Features und ein Target-Feature „Attrition“ mit 1470 Samples

H) deskriptive Datenanalyse:

```
In [9]:  1  df01.describe()
```

Out[9]:

| | Age | DistanceFromHome | JobInvolvement | JobLevel | JobSatisfaction | MonthlyIncome | NumCompaniesWorked | PercentSalaryHike | PerformanceRat |
|---|---|---|---|---|---|---|---|---|---|
| count | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000000 | 1470.000 |
| mean | 36.923810 | 9.192517 | 2.729932 | 2.063946 | 2.728571 | 6502.931293 | 2.693197 | 15.209524 | 3.153 |
| std | 9.135373 | 8.106864 | 0.711561 | 1.106940 | 1.102846 | 4707.956783 | 2.498009 | 3.659938 | 0.360 |
| min | 18.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1009.000000 | 0.000000 | 11.000000 | 3.000 |
| 25% | 30.000000 | 2.000000 | 2.000000 | 1.000000 | 2.000000 | 2911.000000 | 1.000000 | 12.000000 | 3.000 |
| 50% | 36.000000 | 7.000000 | 3.000000 | 2.000000 | 3.000000 | 4919.000000 | 2.000000 | 14.000000 | 3.000 |
| 75% | 43.000000 | 14.000000 | 3.000000 | 3.000000 | 4.000000 | 8379.000000 | 4.000000 | 18.000000 | 3.000 |
| max | 60.000000 | 29.000000 | 4.000000 | 5.000000 | 4.000000 | 19999.000000 | 9.000000 | 25.000000 | 4.000 |

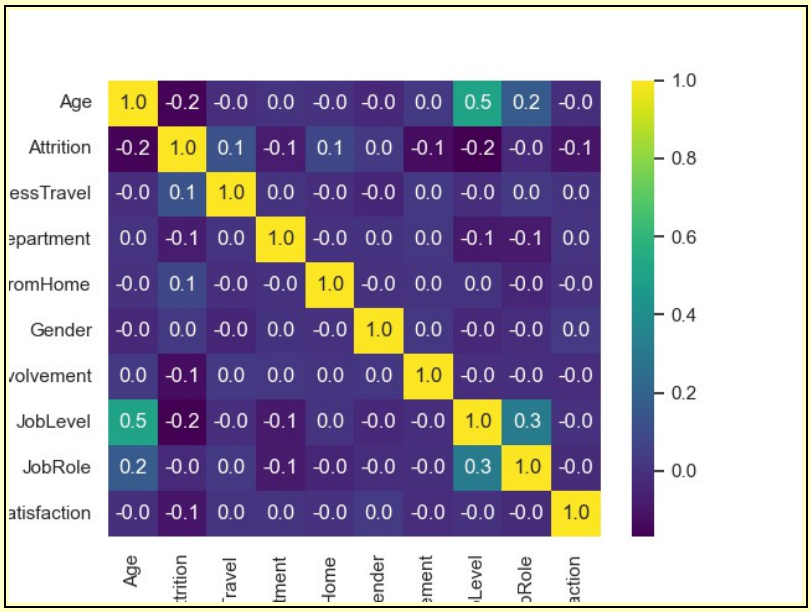
## 4. Korrelationen der Features aufzeigen:

A) df.iloc[::,:10].corr() → numerische Korrelation in Tabellenform

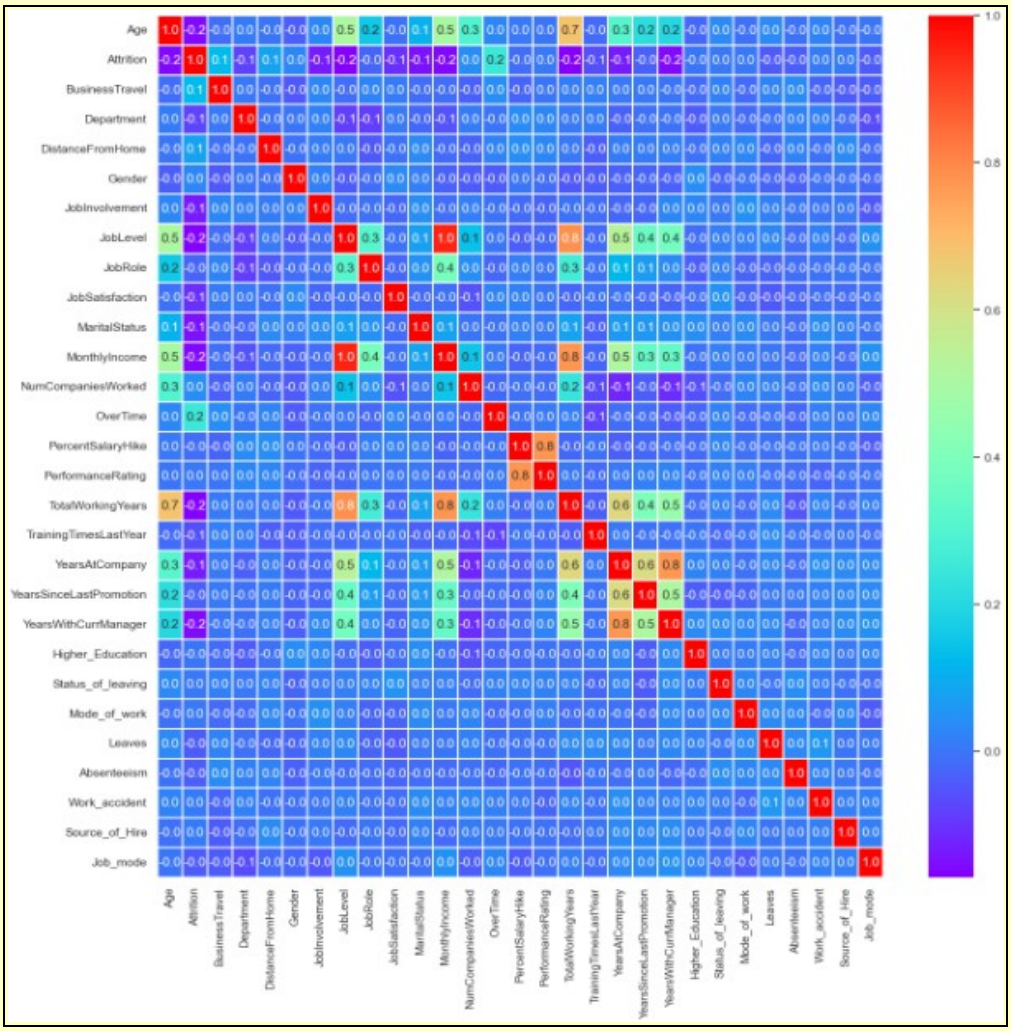
```
1  df02.iloc[::,:10].corr()
```

| | Age | Attrition | BusinessTravel | Department | DistanceFromHome | Gender | JobInvolvement | JobLevel | JobRole | JobSatisfactio |
|---|---|---|---|---|---|---|---|---|---|---|
| Age | 1.000000 | -0.159205 | -0.011807 | 0.007652 | -0.001686 | -0.036311 | 0.029820 | 0.509604 | 0.159715 | -0.00489 |
| Attrition | -0.159205 | 1.000000 | 0.127006 | -0.077351 | 0.077924 | 0.029453 | -0.130016 | -0.169105 | -0.027930 | -0.10348 |
| BusinessTravel | -0.011807 | 0.127006 | 1.000000 | 0.005436 | -0.009696 | -0.044896 | 0.029300 | -0.011696 | 0.023537 | 0.00866 |
| Department | 0.007652 | -0.077351 | 0.005436 | 1.000000 | -0.002196 | 0.000488 | 0.017693 | -0.088018 | -0.084779 | 0.00623 |
| DistanceFromHome | -0.001686 | 0.077924 | -0.009696 | -0.002196 | 1.000000 | -0.001851 | 0.008783 | 0.005303 | -0.043595 | -0.00366 |
| Gender | -0.036311 | 0.029453 | -0.044896 | 0.000488 | -0.001851 | 1.000000 | 0.017960 | -0.039403 | -0.014824 | 0.03325 |
| JobInvolvement | 0.029820 | -0.130016 | 0.029300 | 0.017693 | 0.008783 | 0.017960 | 1.000000 | -0.012630 | -0.006766 | -0.02147 |
| JobLevel | 0.509604 | -0.169105 | -0.011696 | -0.088018 | 0.005303 | -0.039403 | -0.012630 | 1.000000 | 0.313588 | -0.00194 |
| JobRole | 0.159715 | -0.027930 | 0.023537 | -0.084779 | -0.043595 | -0.014824 | -0.006766 | 0.313588 | 1.000000 | -0.02425 |
| JobSatisfaction | -0.004892 | -0.103481 | 0.008666 | 0.006231 | -0.003669 | 0.033252 | -0.021476 | -0.001944 | -0.024258 | 1.00000 |

B) in Form eines Heatmaps mit ausgewählten Features




C) Heatmap mit allen Features



D) Korrelation mit der einzelnen Spalten „Attrition“: corr_target = df.corr()[["Target"]]

| | Attrition | categories |
|---|---|---|
| 0 | -0.159205 | Age |
| 1 | 1.000000 | Attrition |
| 2 | 0.127006 | BusinessTravel |
| 3 | -0.077351 | Department |
| 4 | 0.077924 | DistanceFromHome |
| 5 | 0.029453 | Gender |
| 6 | -0.130016 | JobInvolvement |
| 7 | -0.169105 | JobLevel |
| 8 | -0.027930 | JobRole |
| 9 | -0.103481 | JobSatisfaction |
| 10 | -0.145985 | MaritalStatus |
| 11 | -0.159840 | MonthlyIncome |
| 12 | 0.043494 | NumCompaniesWorked |
| 13 | 0.246118 | OverTime |
| 14 | -0.013478 | PercentSalaryHike |
| 15 | 0.002889 | PerformanceRating |

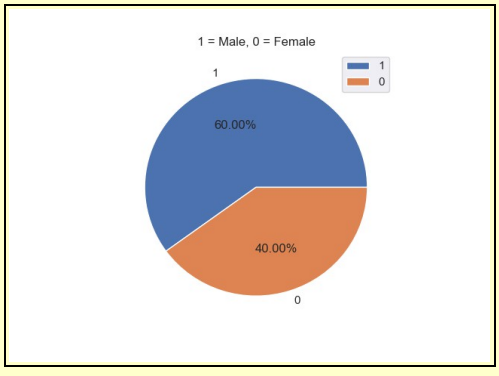
| | | |
|---|---|---|
| 15 | 0.002889 | PerformanceRating |
| 16 | -0.171063 | TotalWorkingYears |
| 17 | -0.059478 | TrainingTimesLastYear |
| 18 | -0.134392 | YearsAtCompany |
| 19 | -0.033019 | YearsSinceLastPromotion |
| 20 | -0.156199 | YearsWithCurrManager |
| 21 | -0.000188 | Higher_Education |
| 22 | 0.005997 | Status_of_leaving |
| 23 | 0.006742 | Mode_of_work |
| 24 | -0.041820 | Leaves |
| 25 | -0.037867 | Absenteeism |
| 26 | 0.009846 | Work_accident |
| 27 | 0.009571 | Source_of_Hire |
| 28 | -0.039706 | Job_mode |

E) höchste Korrelationen mit folgenden Features:

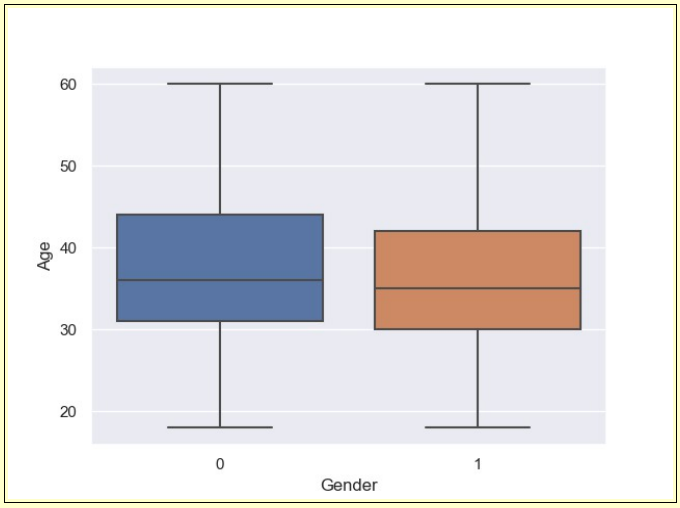
- ✔ 1.0 Korrelation: MonthlyIncome – JobLevel
- ✔ 0.8 Korrelationen: JobLevel – JobWorkingYears, YearsAtCompany –YearsWithCurrentManager, PerformanceRating – PercentSalaryHike, MonthlyIncome – TotalWorkingYears, JobLevel - TotalWorkingYears,
- ✔ 0.7 Korrelationen: Age - TotalWorkingYears,
- ✔ 0.6 Korrelationen: YearsAtCompany – YearsSinceLastPromotion, TotalWorkingYears - YearsAtCompany
- ✔ Attrition Korrelation: 0.2 Korrelation mit OverTime

## 5. Visualisierungen

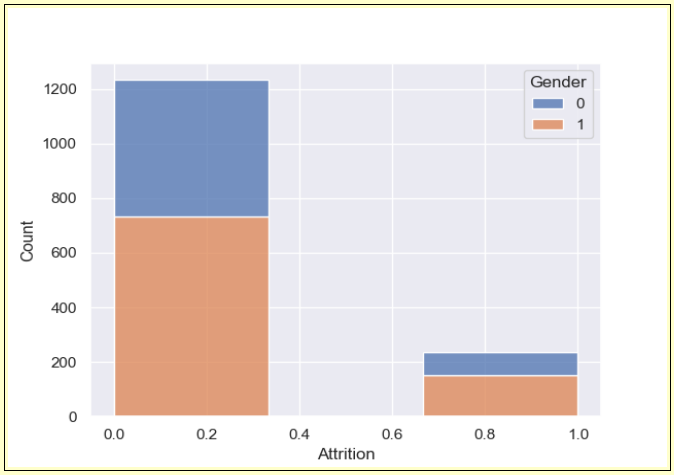
A) Geschlechter-Verteilung:



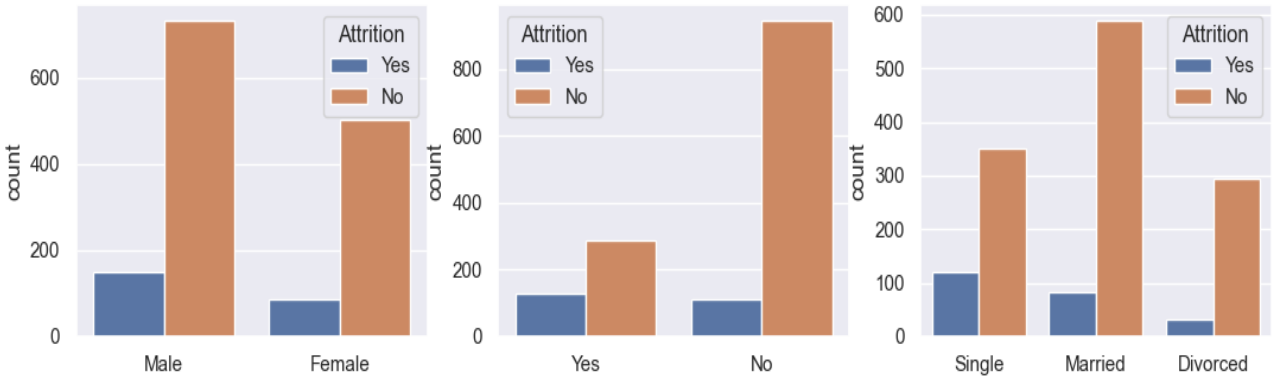
B) Geschlecht-Alter-Verteilung:




C) Kündigung-Geschlecht-Verteilung:




D) Kündigung-Verteilungen:

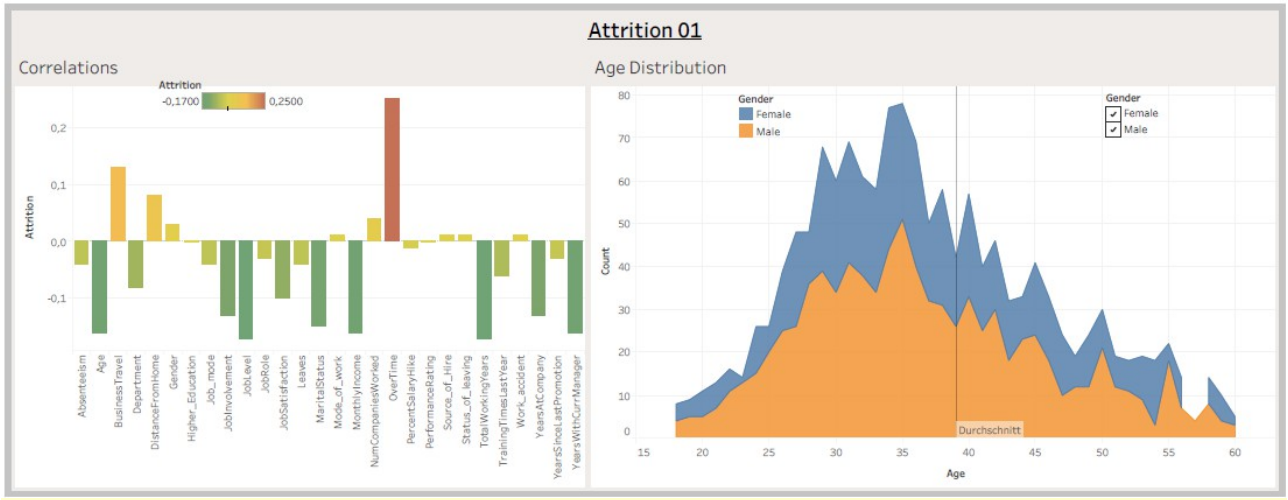


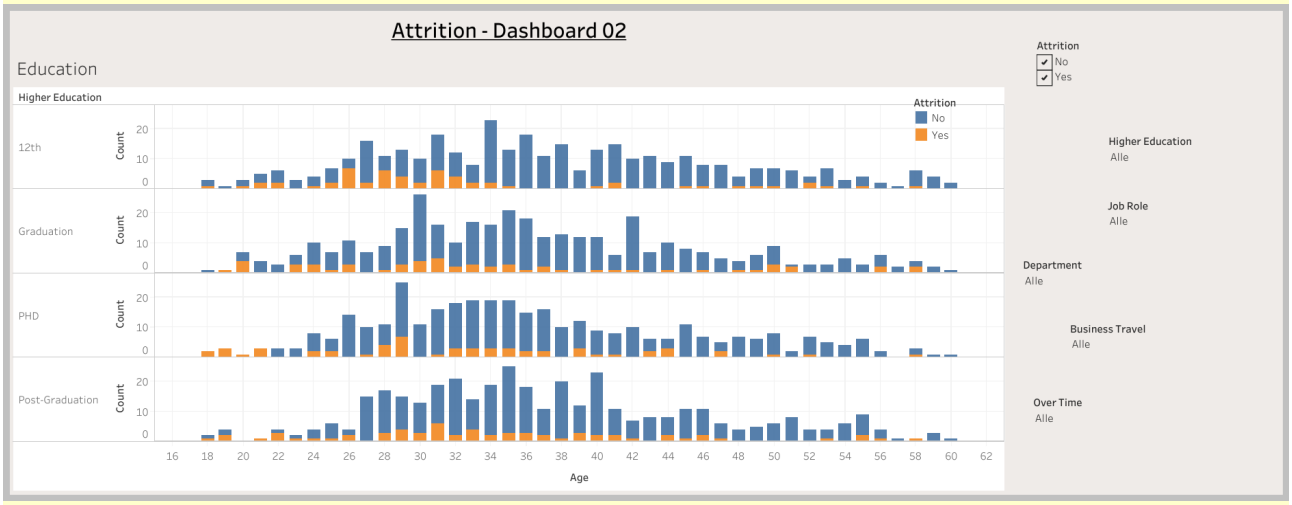
## 6. DataFrame exportieren für weitere Anwendung in Tableau:

A) df.to_csv("....csv",index=False)

B) Presentation der Analyse des Datensatzes erreichbar unter folgendem Link: https://public.tableau.com/app/profile/dietmar.kiendl/viz/Attrition_16969574831890/Attrition01

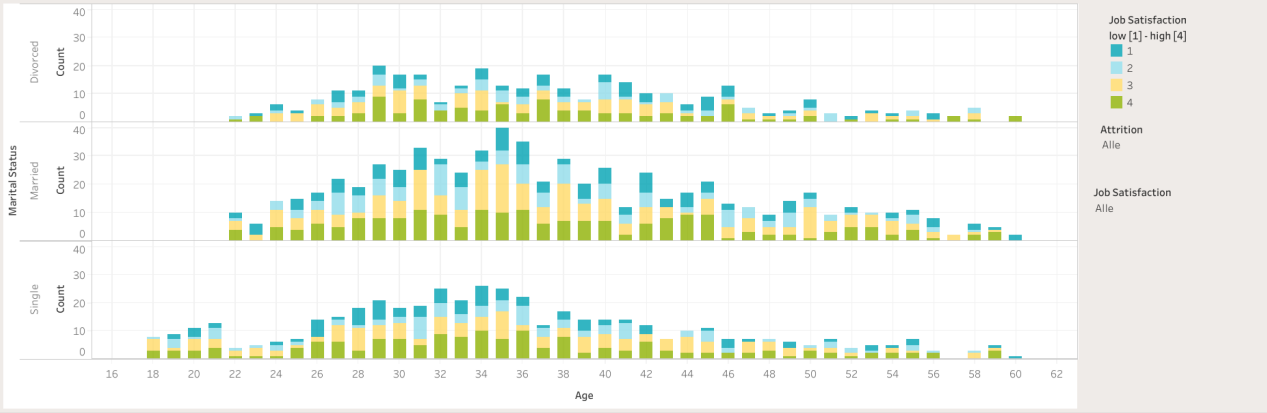
C) der Link ist notwendig, da man sonst die Filter, die in den Dashboards eingebaut sind, nicht für eine spezifische Betrachtungsweise der Diagramme nutzen kann.



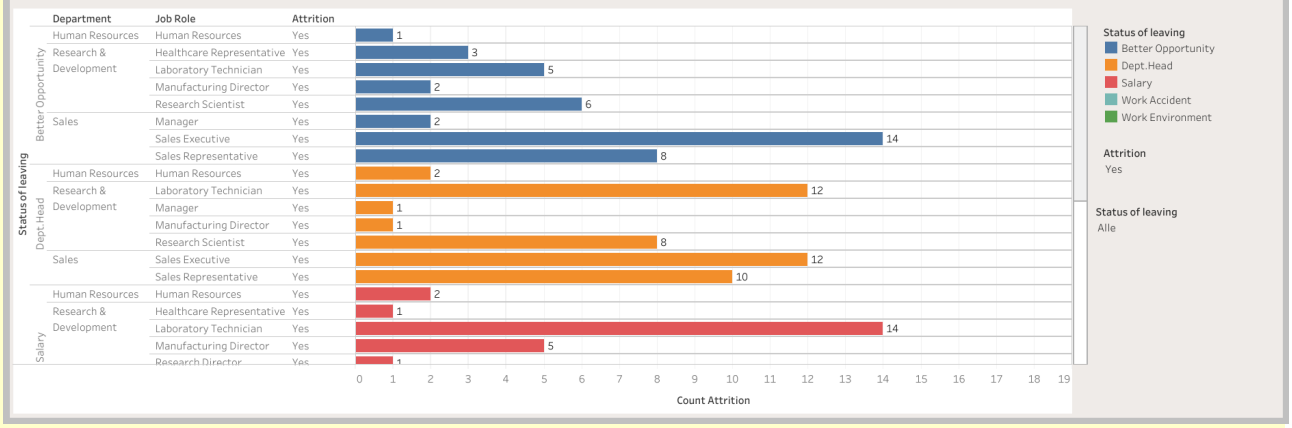
## Attrition - Dashboard 03

Person



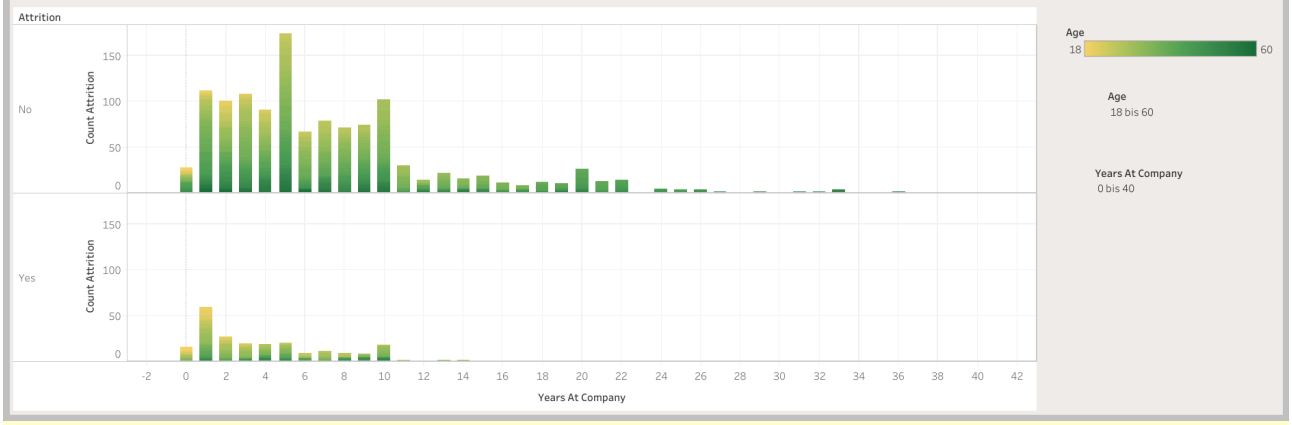

## Attrition - Dashboard 04

Reason



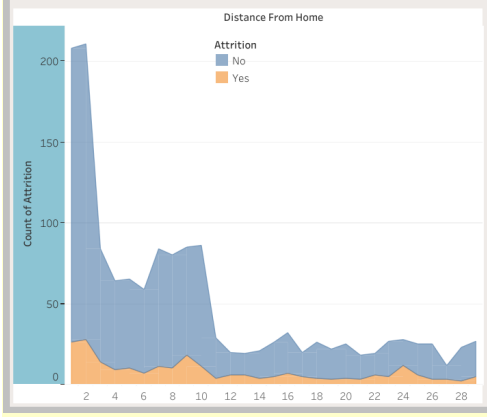

## Attrition - Dashboard 05

Years at Company - Age



## Attrition - Dashboard 06

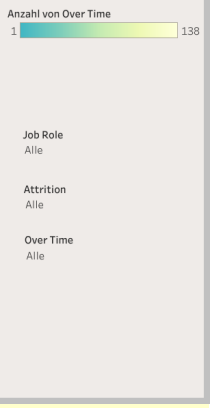
### Conditions01




### Conditions02

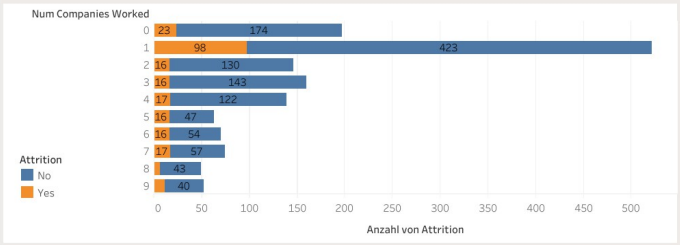
Attrition / Business Travel

| Job Role | Over Time | No | | | Yes | | |
|---|---|---|---|---|---|---|---|
| | | Non-Travel | Travel_Freq.. | Travel_Rarely | Non-Travel | Travel_Freq.. | Travel_Rarely |
| Healthcare Representative | No | 9 | 15 | 63 | 1 | 2 | 4 |
| | Yes | 5 | 8 | 22 | | 1 | 1 |
| Human Resources | No | 4 | 4 | 24 | | 2 | 5 |
| | Yes | | 2 | 6 | | 2 | 3 |
| Laboratory Technician | No | 23 | 29 | 114 | 1 | 9 | 21 |
| | Yes | 3 | 6 | 22 | 1 | 7 | 23 |
| Manager | No | 8 | 10 | 56 | | | 1 |
| | Yes | 3 | 3 | 17 | 1 | | 3 |
| Manufacturing Director | No | 10 | 21 | 69 | | 3 | 3 |
| | Yes | 3 | 5 | 27 | | | 4 |
| Research Director | No | 5 | 5 | 46 | | | 1 |
| | Yes | 1 | 7 | 14 | | | 1 |
| Research Scientist | No | 19 | 25 | 137 | 1 | 3 | 10 |
| | Yes | 5 | 14 | 45 | 3 | 12 | 18 |
| Sales Executive | No | 27 | 41 | 138 | 2 | 6 | 18 |
| | Yes | 8 | 5 | 50 | 2 | 7 | 22 |
| Sales Representative | No | 5 | 8 | 29 | | 8 | 9 |
| | Yes | | | 8 | | 7 | 9 |




## Attrition - Dashboard 07

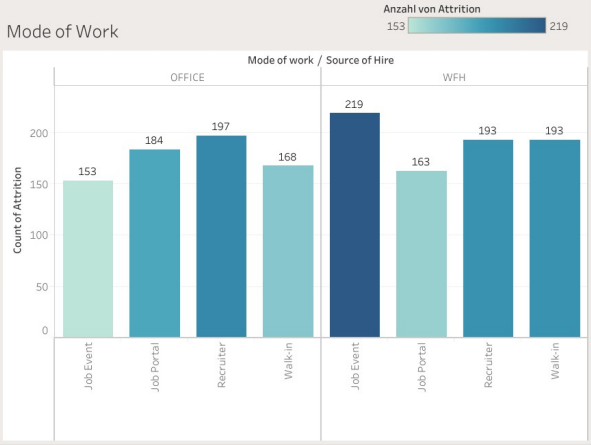
### Working Ethic




Monthly Income
1009 bis 19999

Job Involvement
Alle

### Mode of Work



# . Bauen eines Machine Learning Modells

## 1 train_test_split --> ML, Supervised/Unsupervised Learning

A) Supervised Learning mit allen Features

- ✗ train-/test-daten erstellen → Bibliothek importieren: from sklearn.model_selection import train_test_split

```
# train-/test-daten:
from sklearn.model_selection import train_test_split
```

```
X01 = df02.drop("Attrition",axis=1)
y01 = df02["Attrition"]
```

```
X_train, X_test, y_train, y_test = train_test_split(X01, y01, test_size=0.20, random_state=33)
```

- ✗ X : Train/Test-Daten → alle Features außer "Attrition"
- ✗ y : Train/Test-Daten → nur Feature "Attrition"
- ✗ alle X-daten standardisieren: mit der Bibliothek „StandardScaler“

```
# alle X-daten standardisieren:
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
```

- ✗ alle Bibliotheken mit benötigten Algorithmen importieren:

```
# algorithmen importieren:
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV

#initialisieren:
log = LogisticRegression()
knc = KNeighborsClassifier()
nab = GaussianNB()
rfc = RandomForestClassifier()
svc = SVC()
```

(1) LogisticRegression (am günstigsten für Kündigungs-Klassifizierungen)

(2) KneighborsClassifieror (Elbow)

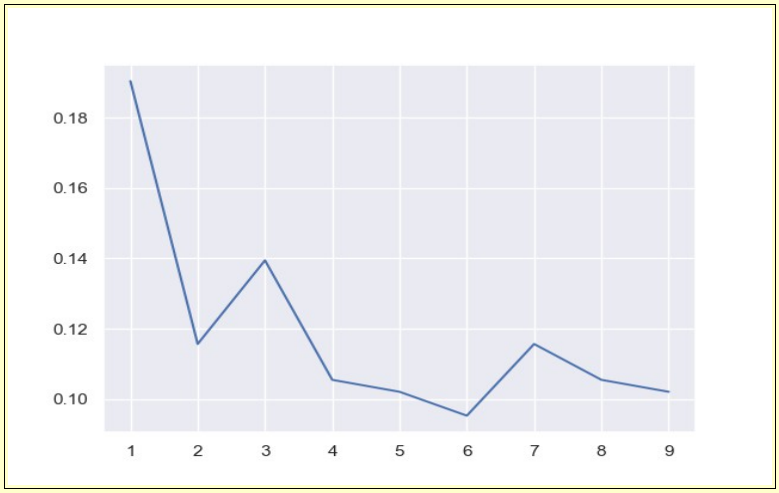
(3) SVC (mit Gridsearch → bestes C/gamma ermitteln)

(4) NaiveBayes

(5) RandomForestClassifier

✗ mit der Elbow-Method das beste k für n_neighbors ermitteln:

```
# elbow method:
fehler = []
for k in range(1,10):
    knc = KNeighborsClassifier(n_neighbors=k)
    knc.fit(X_train,y_train)
    predictions_k = knc.predict(X_test)
    fehler.append(np.mean(predictions_k != y_test))
plt.plot(range(1,10),fehler)
#                                wählen  --> k = 2!  ---> k = 6 am besten!!
```




✗ beste werte für C und gamma (SVC, Hyperebenen) ermitteln

```
#beste werte für C und gamma (SVC,hyperebenen):
hyper_para = {"C":[0.1,1,10,100,1000],"gamma":[1,0.1,0.01,0.001,0.0001]}
# initialisieren von Gridsearch
grid = GridSearchCV(SVC(),hyper_para,refit=True)
grid.fit(X_train,y_train)
```

```
grid.best_params_
```

```
{'C': 1000, 'gamma': 0.0001}
```

- ✗ Initialisieren und Trainieren: log = LogisticRegression().fit(X_train,y_train) und auch mit den anderen 4 Algorithmen, um sie zu vergleichen

```
# neu initialisieren nach "elbow" und "C/gamma" und trainieren:
log = LogisticRegression().fit(X_train,y_train)
knc = KNeighborsClassifier(n_neighbors=6).fit(X_train,y_train)
nab = GaussianNB().fit(X_train,y_train)
rfc = RandomForestClassifier(n_estimators=1000,random_state=33).fit(X_train,y_train)
svc = SVC(C=1000,gamma=0.0001).fit(X_train,y_train)
```

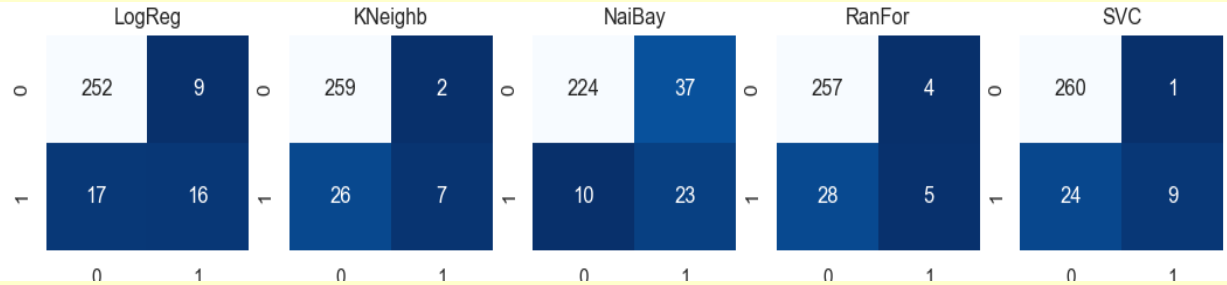
- ✗ Vorhersagen für alle 5 Alg.: pred_log = log.predict(X_test)

```
# vorhersagen:
pred_log = log.predict(X_test)
pred_knc = knc.predict(X_test)
pred_nab = nab.predict(X_test)
pred_rfc = rfc.predict(X_test)
pred_svc = svc.predict(X_test)
```

- ✗ Bibliotheken importieren, um die Accuracy, Precision, Recall, etc. des jeweiligen Algorithmus zu ermitteln → bester Algorithmus

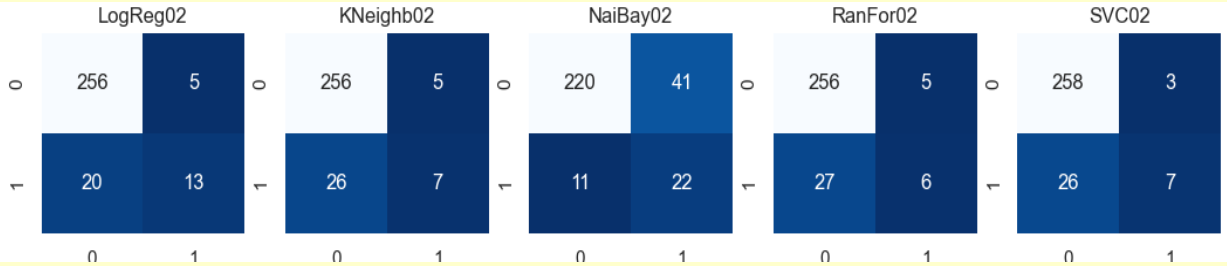
```
from sklearn.metrics import accuracy_score,confusion_matrix,mean_absolute_error,classification_report
```

✗ Confusion-Matrix mit allen Features:



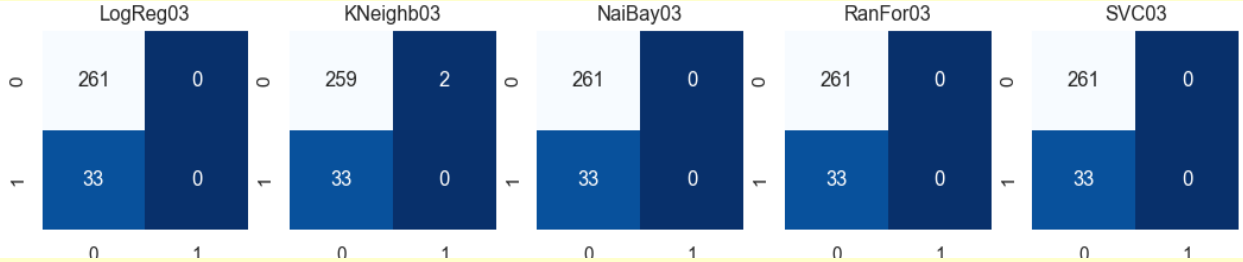

B) Supervised Learning mit Features mit höchsten Korrelationen

- Confusion-Matrix:




C) Supervised Learning mit Features mit niedrigsten Korrelationen

✗ Confusion-Matrix:

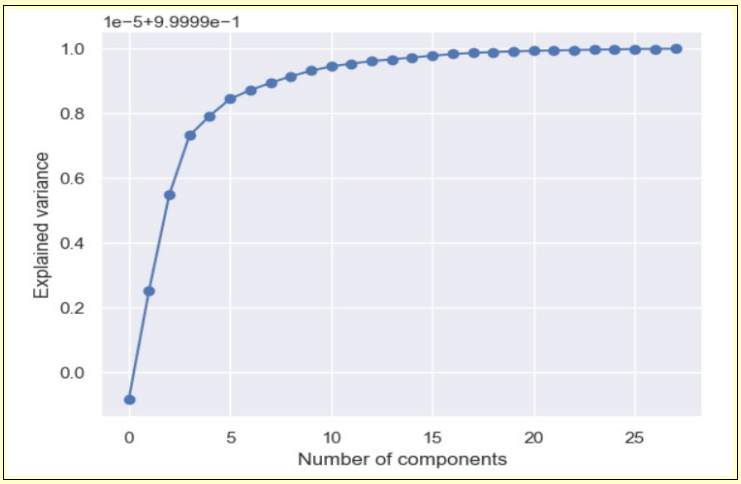


D) Supervised Learning mit Features reduziert durch PCA

- ✗ Überprüfung mit ursprünglichem datensatz, wie viele components tatsächlich wichtig sind: → der Datensatz läßt sich sinnvollerweise auf ein eiinziges Feature reduzieren, wie die Graphik zeigt:

```
# reduzieren der daten mit PCA:
from sklearn.decomposition import PCA
```

```
# überprüfung mit ursprünglichem datensatz, wie viele components wichtig sind:
pca_check = PCA(n_components=28,random_state=33)
pca_check.fit(df02)
plt.plot(np.cumsum(pca_check.explained_variance_ratio_),marker="o")
plt.xlabel("Number of components")
plt.ylabel("Explained variance")
plt.show()
```




- ✗ auch die Berechnung bestätigt es: mit .explained_variance_ratio_ ermittle ich die Werte der notwenigen Varianz:

```
for i,value in enumerate(pca_check.explained_variance_ratio_):
    print(f"{i+1}. Principal Component erklärt {value*100:.4f}% der Varianz ")
```

```
1. Principal Component erklärt 99.9989% der Varianz
2. Principal Component erklärt 0.0003% der Varianz
3. Principal Component erklärt 0.0003% der Varianz
4. Principal Component erklärt 0.0002% der Varianz
5. Principal Component erklärt 0.0001% der Varianz
```

✗ eine Heatmap zeigt den Datensatz mit nur mehr einem Feature, aber großer Varianz:

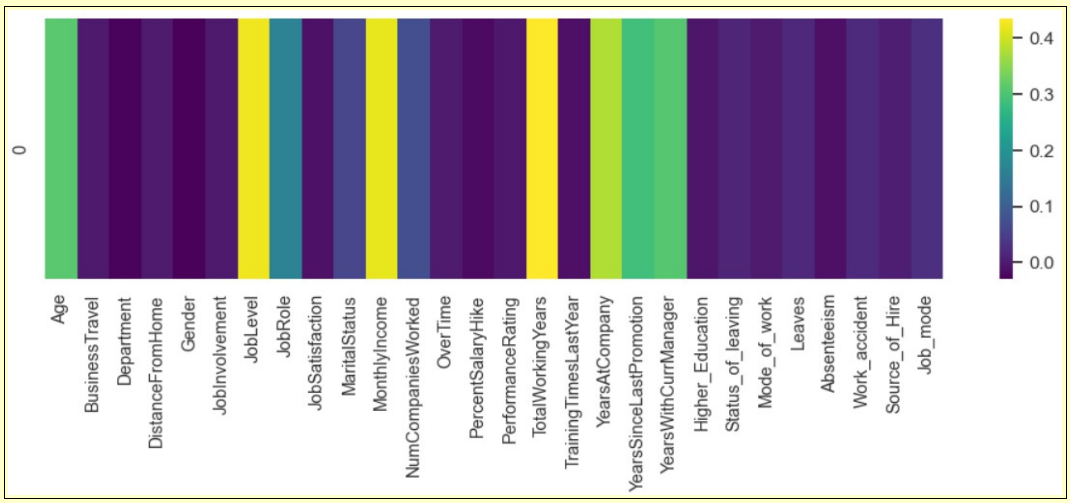
```
# 1 principal component --> initialisieren:
pca = PCA(n_components=1,random_state=33)
```

```
# trainieren und transformieren:
x_pca = pca.fit_transform(scaled_X04)
```

```
print(scaled_X04.shape)
print(x_pca.shape)
print(type(x_pca))
```

```
(1470, 28)
(1470, 1)
<class 'numpy.ndarray'>
```

✗ Gewichtungen der ursprünglich 28 Principal Components in einer einzigen:



## 2 Confusion-Matrix-Gesamt-Vergleich:

```
Alle Korrelationen:
[[252   9]
 [ 17  16]] LogReg
--------------------------------
[[259   2]
 [ 26   7]] KNeighbor
--------------------------------
[[224  37]
 [ 10  23]] NaiveBayes
--------------------------------
[[257   4]
 [ 28   5]] RandomForest
--------------------------------
[[260   1]
 [ 24   9]] SVC
--------------------------------
```

```
Höchste Korrelationen:
[[256   5]
 [ 20  13]] LogReg
--------------------------------
[[256   5]
 [ 26   7]] KNeighbor
--------------------------------
[[220  41]
 [ 11  22]] NaiveBayes
--------------------------------
[[256   5]
 [ 27   6]] RandomForest
--------------------------------
[[258   3]
 [ 26   7]] SVC
--------------------------------
```

```
Niedrigste Korrelationen:
[[261   0]
 [ 33   0]] LogReg
--------------------------------
[[259   2]
 [ 33   0]] KNeighbor
--------------------------------
[[261   0]
 [ 33   0]] NaiveBayes
--------------------------------
[[261   0]
 [ 33   0]] RandomForest
--------------------------------
[[261   0]
 [ 33   0]] SVC
--------------------------------
```

```
PCA reduziert:
[[261   0]
 [ 33   0]] LogReg
--------------------------------
[[255   6]
 [ 29   4]] KNeighbor
--------------------------------
[[261   0]
 [ 33   0]] NaiveBayes
--------------------------------
[[210  51]
 [ 26   7]] RandomForest
--------------------------------
[[257   4]
 [ 29   4]] SVC
--------------------------------
```

## 3 Accuracy-Score-Gesamt-Vergleich:

```
Alle Korrelationen:
LogReg         :      91.16 %
KNeighbor      :      90.48 %
NaiveBayes     :      84.01 %
RandomForest   :      89.12 %
SVC            :      91.50 %
```

```
Höchste Korrelationen:
LogReg02       :      91.50 %
KNeighbor02    :      89.46 %
NaiveBayes02   :      82.31 %
RandomForest02:       89.12 %
SVC02          :      90.14 %
```

```
Niedrigste Korrelationen:
LogReg03       :      88.78 %
KNeighbor03    :      88.10 %
NaiveBayes03   :      88.78 %
RandomForest03:       88.78 %
SVC03          :      88.78 %
```

```
PCA reduziert:
LogReg05       :      88.78 %
KNeighbor05    :      88.10 %
NaiveBayes05   :      88.78 %
RandomForest05:       73.81 %
SVC05          :      88.78 %
```

## 4 Schlussfolgerung → bester Logarithmus

A) alle Features werden benutzt, man erhält hier die besten Werte

| | accuracy | precision | recall | error |
|---|---|---|---|---|
| **LogisticRegression** | 0.91 | 0.97 | 0.94 | 0.09 |
| **KNeighborsClassifier** | 0.90 | 0.99 | 0.91 | 0.10 |
| **NaiveBayes** | 0.84 | 0.86 | 0.96 | 0.16 |
| **RandomForestClassifier** | 0.89 | 0.98 | 0.90 | 0.11 |
| **SVC** | 0.91 | 1.00 | 0.92 | 0.09 |

B) unter Berücksichtigung aller Parameter schneidet am besten ab und wird somit ausgewählt: → SVC (Support Vector Machine)

C) die am besten berrechneten Hyperparameter sind C=1000 und gamma=0.0001

D) der Classification-Report liefert folgende Ergebnisse:

SVC

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.92 | 1.00 | 0.95 | 261 |
| 1 | 0.90 | 0.27 | 0.42 | 33 |
| accuracy | | | 0.91 | 294 |
| macro avg | 0.91 | 0.63 | 0.69 | 294 |
| weighted avg | 0.91 | 0.91 | 0.89 | 294 |

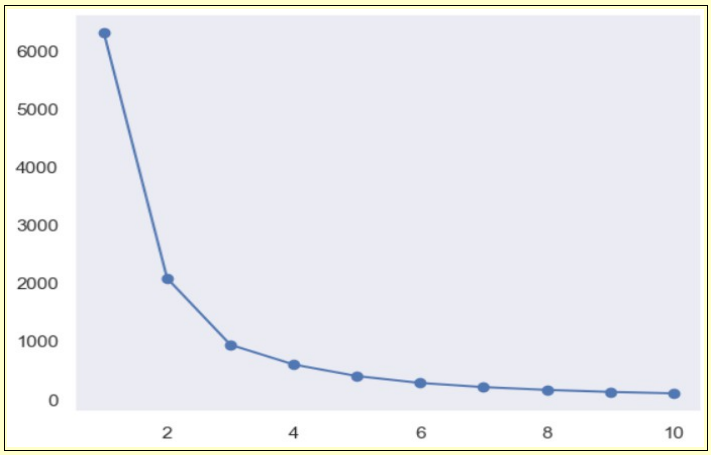
## 5 Versuch, ob man mit einem Algorithmus des Unsupervised Learning ebenfalls auf gute Werte bringen kann, scheiterte:

A) gewählt wurde der KMeans_Clustering-Algorithmus

B) normalerweise berechnet man die Anzahl von n_cluster über die .inertias_ (Distanzen bzw. Fehler). Auch wenn hier die „2“ das günstigste Ergebnis ist, so brauche ich sowieso diesen Wert, weil ich ansonsten nicht mit dem Ergebnis des Features „Attrition“ vergleichen könnte.

```
#             --> inertias (distanzen bzw. fehler)
kmeans_k02 = [KMeans(n_clusters=k, random_state=33).fit(X03) for k in range(1, 11)]
inertias02 = [model.inertia_ for model in kmeans_k02]
```

```
# visualisieren:
plt.grid()
plt.plot(range(1, 11),inertias,"bo-")
```

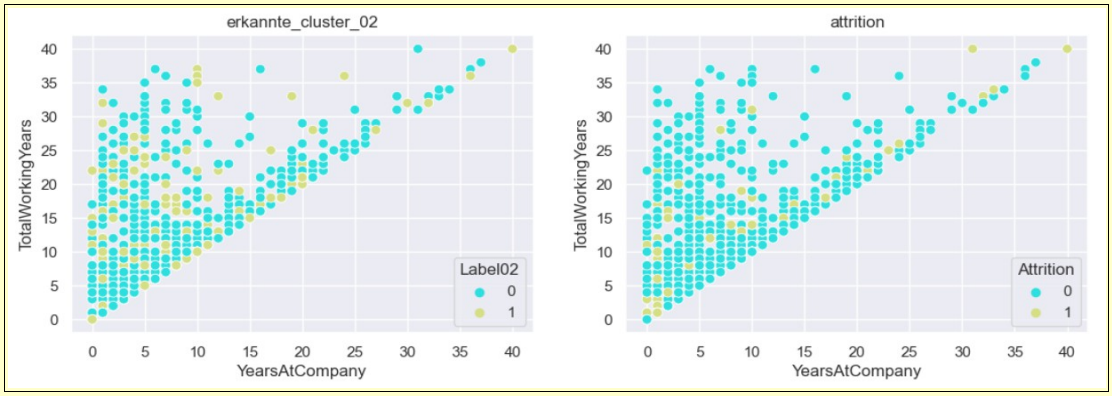



C) Der Classification-Report liefert keine guten Ergebnisse. Zunächst habe ich den mit PCA reduzierten Datensatz benutzt und danach den Datensatz mit den niedrigsten Korrelationen, wie es für den KMeans_Clustering-Algorithmus normalerweise sinnvoll ist. Daraus ergaben sich zwar verbesserte Werte, allerdings keine vergleichbar guten wie beim SVC-Algorithmus.

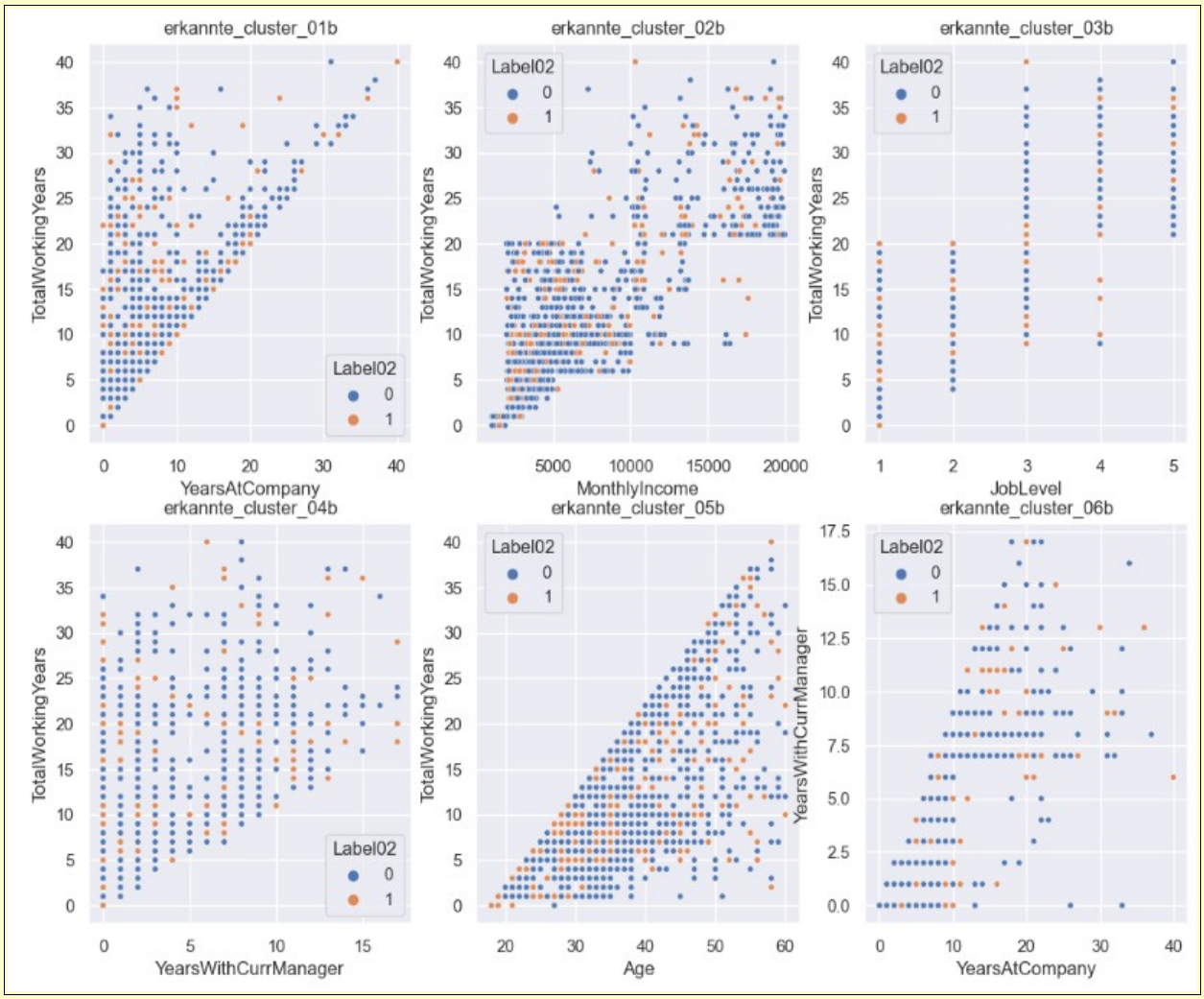
D) Classification-Report mit den konkreten Werten:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.85 | 0.76 | 0.80 | 1233 |
| 1 | 0.20 | 0.32 | 0.25 | 237 |
| accuracy | | | 0.69 | 1470 |
| macro avg | 0.53 | 0.54 | 0.53 | 1470 |
| weighted avg | 0.75 | 0.69 | 0.71 | 1470 |

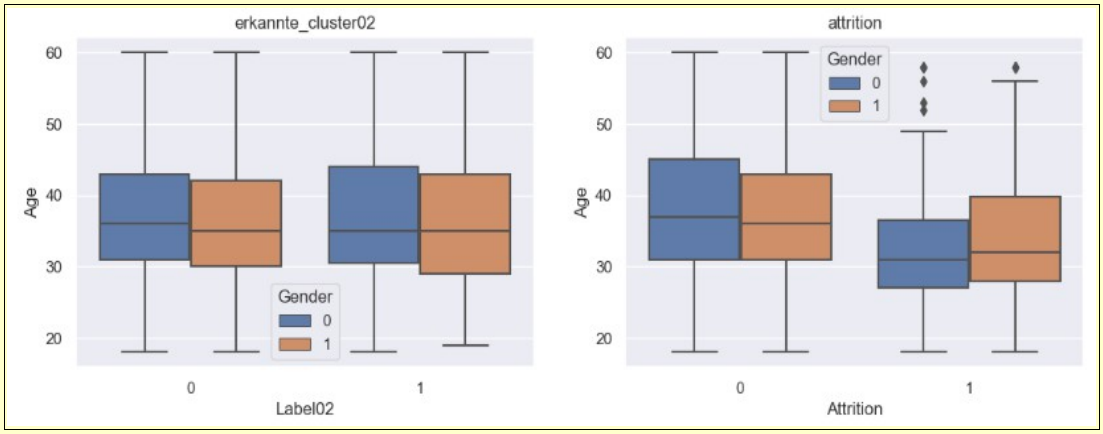
E) ein direkter Vergleich mit dem Target „Attrition“ aus dem Datensatz bestätigt dies:




F) auch bei Vergleichen unter mehreren Features im Scatterplot zeigt sich kein wirklich klares Clustering:



G) bei einem BoxPlot von Age-Gender-Attrition ist das Problem auch sichtbar:




H) Schlussfolgerung: Man kann also ganz deutlich erkennen, dass in diesem Fall der KMeans_Clustering-Algorithmus nicht sinnvoll zur Anwendung kommen kann. Die Gründe hierfür sind:

- ✗ dasTarget benötigt eine Klassifizierung mit nur 2 Werten 0 und 1, beim KMeans_Clustering liegt die Stärke aber in einer Art Segmentierung von Datenbereichen (bei Bildern, Dokumenten, Kunden, Produkten, etc.)
- ✗ auch Ausreißer oder Anomalien spielen in unserem Datensatz keine Rolle, für welche der KMeans_Clustering-Algorithmus normalerweise effektiv eingesetzt werden kann.

## . Gesamtfazit:

- überschaubare Anzahl an Features ließ eine gute Bearbeitung und Analyse des Datensatzes zu
  - keine NULL-Werte
  - ein bereits vorhandenes Label „Attrition“
- beste Ergebnisse in Supervised Learning:
  - mit allen Features (X01)
  - mit den Features mit den höchsten Korrelationen (X02)
- beste Ergebnisse bezüglich der Wahl des richtigen Algorithmus:
  - SVC [→ 91.50 % (X01) Accuracy]
  - Logistische Regression [→ 91.50 % (X02) Accuracy]
- minimal schlechtere Ergebnisse mit den weiteren Algorithmen
  - KneighborsClassifier
  - NaiveBayes
  - RandomForestClassifier
- Ergebnisse mit Features mit den geringsten Korrelationen (X03):
  - mittelmäßig/gut

## . Zukunft:

1. Vernetzung:
   - Bei der den Unternehmen gibt es über API-Schnittstellen die Möglichkeit, die Datensätze zu updaten, wodurch man das ML-Modell auch wieder optimieren kann
2. Automatisierung:
   - Im ML-Modell implementiere ich Methoden, über welche ich den optimalen Algorithmus ermitteln kann, wodurch das ML-Modell genauere Vorhersagen treffen kann.