Stress im Beruf

3. Dezember 2023

1 Einleitung

Die Auswahl eines zur Analyse geeigneten Datensatzes fiel zu Beginn schwer, da keine Erfahrungen im Bereich der Datenanalyse vorlagen und Eigenschaften verschiedener Datensätze nicht adäquat eingeschätzt werden konnten. Nach längerer Überlegung fiel die Entscheidung auf den Datensatz Are Your Employees Burning Out? von kaggle, da das Thema Mental Health im Arbeitsleben immer mehr an Bedeutung gewinnt. Der Datensatz stammt ursprünglich aus einem Wettbewerb des Unternehmens Hackerearth. Bei der Suche nach geeigneten Fragestellungen, die weder zu kompliziert noch zu einfach sind, wurde sich letztlich für die folgenden Drei entschieden:

1. Wie wirkt sich die Dauer der Betriebszugehörigkeit auf die Burnout-Rate unter den Mitarbeitern des Unternehmens aus?
2. Wie wirkt sich der Wert für geistige Erschöpfung auf die Burnout-Rate aus?
3. Haben Angestellte mit Homeoffice (WFH Setup Available) tendenziell eine niedrigere Burnout-Rate als Angestellte ohne Homeoffice?

2 Datensatz einlesen und verstehen

2.1 Bibliotheken zur Datenanalyse und Visualisierung importieren

```
# Bibliotheken zur Datenmanipulation und -analyse
import pandas as pd
# Visualisierung der Daten
import matplotlib.pyplot as plt, seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_absolute_error
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.neighbors import KNeighborsRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.svm import LinearSVR, SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
```

2.2 Datensatz einlesen

```
stress_data = pd.read_csv('data.csv', engine='python' ,parse_dates=['Date of Joining'])
```

2.3 Einen Einblick in den Datensatz erhalten und diesen verstehen

Anhand der Beschreibung des Datensatzes auf kaggle lässt sich diese Liste an Datenpunkten ableiten.

- Employee ID: The unique ID allocated for each employee (example: fffe390032003000)
- Date of Joining: The date-time when the employee has joined the organization (example: 2008-12-30)
- Gender: The gender of the employee (Male/Female)
- Company Type: The type of company where the employee is working (Service/Product)

- WFH Setup Available: Is the work from home facility available for the employee (Yes/No)
- Designation: The designation of the employee of work in the organization. In the range of [0.0, 5.0] bigger is higher designation.
- Resource Allocation: The amount of resource allocated to the employee to work, ie. number of working hours. In the range of [1.0, 10.0] (higher means more resource)
- Mental Fatigue Score: The level of fatigue mentally the employee is facing.
- In the range of [0.0, 10.0] where 0.0 means no fatigue and 10.0 means completely fatigue.
- Burn Rate: The value we need to predict for each employee telling the rate of Bur out while working.
- In the range of [0.0, 1.0] where the higher the value is more is the burn out.

In der Tabelle von Hackerearth lassen sich detailliertere Beschreibungen zu den einzelnen Werten finden.

Variable Description

Column Name	Description
Employee ID	Unique Id of the employee
Date of Joining	Date on which the employee joined the company
Gender	Gender of the employee
Company Type	Type of company eg: Service based, product based etc.
WFH Setup Available	Whether proper work from home setup is available or not
Designation	Seniority level of the employee in codes
Resource Allocation	Hours allocated per day
Mental Fatigue Score	Stress rating provided by employees
Burn Rate	Rate of saturation or burn out rate [Target]

Erste Einblicke in den Datensatz:

```
# Anzahl der Zeilen im DataFrame
len(stress_data)
```

```
22750
```

```
# Anzeigen der ersten Zeilen des DataFrames
stress_data.head()
```

	Employee ID	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	fffe32003000360033003200	2008-09-30	Female	Service	No	2.0	3.0	3.8	0.16
1	fffe3700360033003500	2008-11-30	Male	Service	Yes	1.0	2.0	5.0	0.36
2	fffe31003300320037003900	2008-03-10	Female	Product	Yes	2.0	NaN	5.8	0.49
3	fffe32003400380032003900	2008-11-03	Male	Service	Yes	1.0	1.0	2.6	0.20
4	fffe31003900340031003600	2008-07-24	Female	Service	No	3.0	7.0	6.9	0.52

Der Datensatz hat 22750 Zeilen, wovon die ersten 5 tabellarisch abgebildet wurden. Dabei liegt bei der Ressource Allocation in der mit 2 nummerierten Zeile ein NaN-Wert vor. Dieser und alle weiteren NaN-Werte müssen entfernt werden, um Verfälschungen bei der Analyse auszuschließen.

3 Datensatz vorbereiten

Die NaN-Werte werden zunächst gezählt und entfernt.

```
# Überprüfen auf fehlende Werte in den Spalten
stress_data.isnull().sum()
```

```
Employee ID                0
Date of Joining            0
Gender                     0
Company Type               0
WFH Setup Available        0
Designation                0
Resource Allocation     1381
Mental Fatigue Score    2117
Burn Rate               1124
dtype: int64
```

```
# Entfernen von Zeilen mit fehlenden Werten
stress_data = stress_data.dropna()
len(stress_data)
```

```
18590
```

Jetzt sind von den ursprünglichen 22750 Zeilen noch 18590 übrig. Nun wird die Spalte Employee ID entfernt, da sie für die Datenanalyse nicht relevant ist. Die ersten 5 Zeilen werden erneut in Tabellenform angezeigt:

```
# Entfernen der Spalte 'Employee ID'
stress_data = stress_data.drop("Employee ID", axis=1)
stress_data.head()
```

	Date of Joining	Gender	Company Type	WFH Setup Available	Designation	Resource Allocation	Mental Fatigue Score	Burn Rate
0	2008-09-30	Female	Service	No	2.0	3.0	3.8	0.16
1	2008-11-30	Male	Service	Yes	1.0	2.0	5.0	0.36
3	2008-11-03	Male	Service	Yes	1.0	1.0	2.6	0.20
4	2008-07-24	Female	Service	No	3.0	7.0	6.9	0.52
5	2008-11-26	Male	Product	Yes	2.0	4.0	3.6	0.29

Um sicher zu stellen, dass alle NaN-Werte entfernt wurden, werden die entsprechenden Informationen zum verbliebenen Datensatz abgerufen.

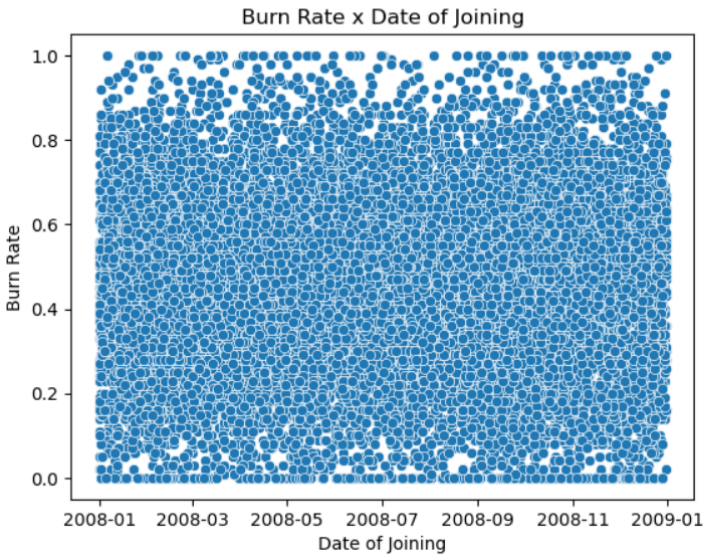
```
# Anzeigen von Informationen über das DataFrame, einschließlich Datentypen
stress_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 18590 entries, 0 to 22749
Data columns (total 8 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Date of Joining       18590 non-null  datetime64[ns]
 1   Gender                18590 non-null  object
 2   Company Type          18590 non-null  object
 3   WFH Setup Available   18590 non-null  object
 4   Designation           18590 non-null  float64
 5   Resource Allocation   18590 non-null  float64
 6   Mental Fatigue Score  18590 non-null  float64
 7   Burn Rate             18590 non-null  float64
dtypes: datetime64[ns](1), float64(4), object(3)
memory usage: 1.3+ MB
```

4 Datensatz analysieren

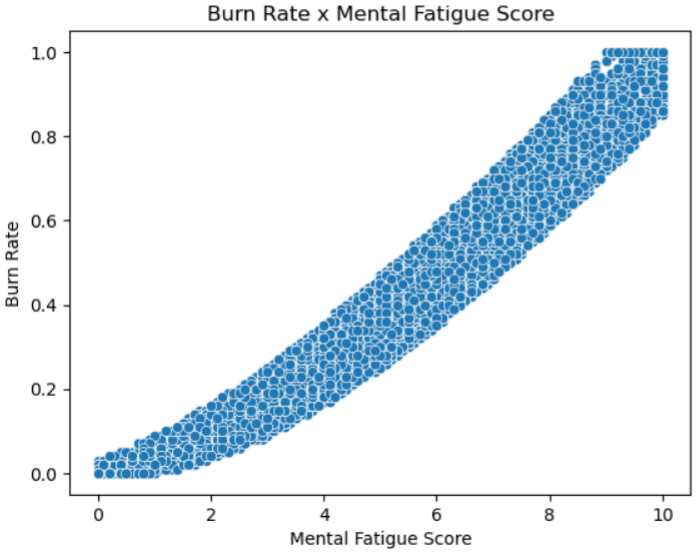
Zu Beginn werden die Fragen der Einleitung beantwortet. Zur Beantwortung der ersten Frage, wie sich die Dauer der Betriebszugehörigkeit auf die Burnout-Rate auswirkt, werden Date of Joining und Burn Rate gegenüber gestellt.

```
visualize_scatter(stress_data, 'Date of Joining', 'Burn Rate', title='Burn Rate x Date of Joining')
```



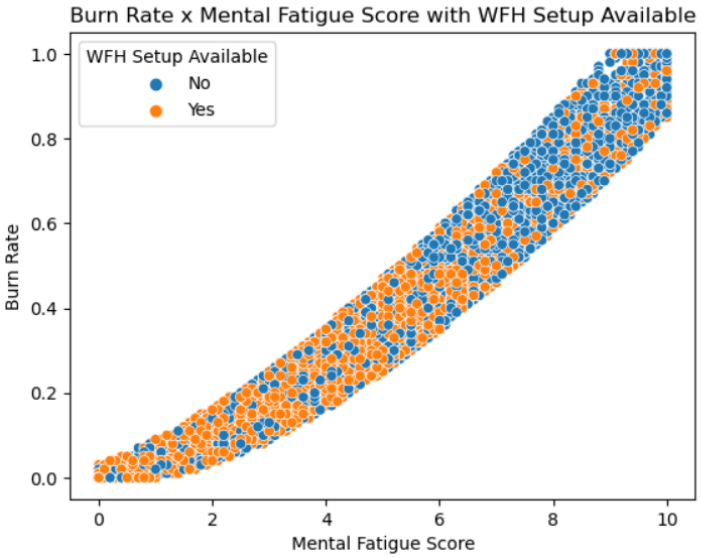
In dieser Graphik lässt sich kein Zusammenhang zwischen dem Date of Joining und der Burn Rate feststellen. Das könnte aber auch daran liegen, dass im Datensatz die Date of Joining Werte alle innerhalb eines Jahres liegen. Der Datensatz erfasst deshalb nur Mitarbeiter, welche innerhalb eines Jahres dem Unternehmen beigetreten sind, wodurch sich anhand dieser Daten keine abschließende Aussage über einen kausalen Zusammenhang zwischen Dauer der Betriebszugehörigkeit und Burn Rate treffen lässt. Die Vorgehensweise zu Frage 2 ist analog:

```
visualize_scatter(stress_data, 'Mental Fatigue Score', 'Burn Rate', title='Burn Rate x Mental Fatigue Score')
```



Zwischen dem Mental Fatigue Score und der Burn Rate ist ein eindeutiger Zusammenhang erkennbar. Da der Wert WFH Setup Available den Datentyp object hat, lässt sich durch eine farbige Gliederung in WHF Setup Available yes oder no eine deutliche Visualisierung erzielen.

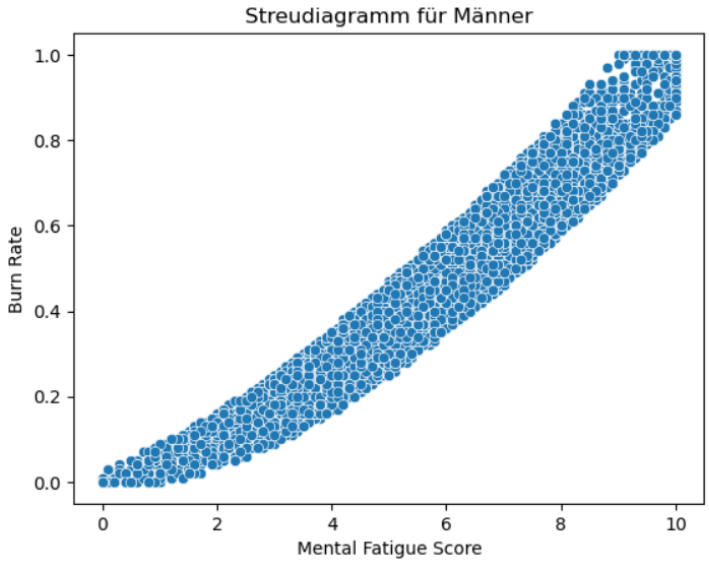
```
visualize_scatter(stress_data, 'Mental Fatigue Score', 'Burn Rate', hue='WFH Setup Available', title='Burn Rate x
```



Hier ist deutlich zu erkennen, dass im linken unteren Bereich deutlich mehr orangene Punkte als im oberen rechten Bereich sind. Die vorliegende Korrelation lässt vermuten, dass Homeoffice eine sehr gute Möglichkeit ist Stress zu reduzieren und einen Burnout zu verhindern. Da die Analyse der Fragestellungen nicht sehr komplex war, wird als nächstes der Zusammenhang zwischen Burn Rate und Mental Fatigue Score näher betrachtet. Dazu werden als Erstes zwei eperate Streudiagramme zu männlichen und weiblichen Personen erstellt.

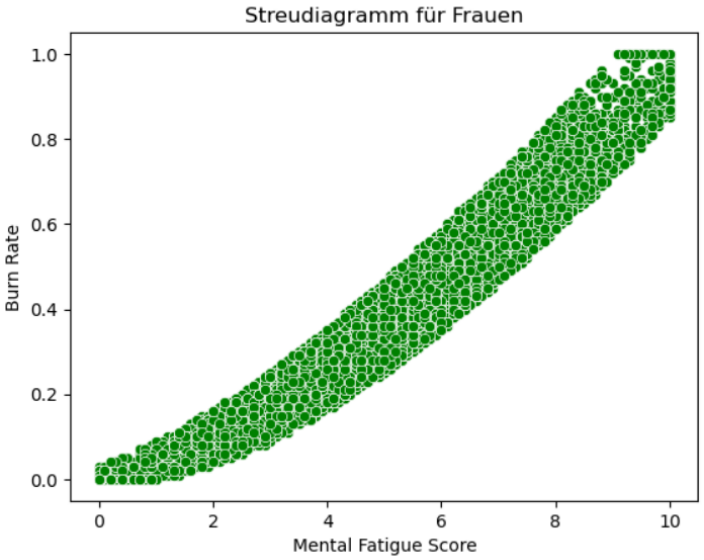
```
male_data = stress_data[stress_data['Gender'] == 'Male']

# Streudiagramm für Männer
visualize_scatter(male_data, 'Mental Fatigue Score', 'Burn Rate', title='Streudiagramm für Männer')
```



```
female_data = stress_data[stress_data['Gender'] == 'Female']

# Streudiagramm für Frauen
visualize_scatter(female_data, 'Mental Fatigue Score', 'Burn Rate', color='green', title='Streudiagramm für Fraue
```



Es ist erkennbar, dass die beiden Streudiagramme sich sehr ähneln, aber nicht komplett identisch sind. Zur weiteren Analyse werden verschiedene Regressionsmodelle angewandt. Dazu werden zunächst einige Methoden definiert:

```
# Funktion zur Visualisierung von Streudiagrammen
def visualize_scatter(data, x, y, hue=None, color=None, title=None):
    sns.scatterplot(data=data, x=x, y=y, hue=hue, color=color)
    plt.title(title)
    plt.show()

# Funktion zur Visualisierung von Streudiagrammen mit Linearer Regression
def visualize_scatter_with_regression(data, x, y, color, title):
    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=x, y=y, data=data, color=color)
    sns.regplot(x=x, y=y, data=data, scatter=False, color='red', line_kws={'label': 'Lineare Regression'})
    plt.title(title)
    plt.xlabel(x)
    plt.ylabel(y)
    plt.show()

# Funktion zum Binarisieren bestimmter Spalten
def binarize_columns(data, columns):
    return pd.get_dummies(data, columns=columns, drop_first=True)


# Funktion zur Bewertung des Modells
def evaluate_model(y_true, y_pred):
    r2 = r2_score(y_true, y_pred)
    mae = mean_absolute_error(y_true, y_pred)
    return r2, mae
```

Daraufhin werden die Regressionen durchgeführt und das Ergebnis in einem neuen Dictionary gespeichert:

```
# Entfernen der Spalte 'Date of Joining'
stress_data = stress_data.drop('Date of Joining', axis=1)

# Binarisierung
binary_cols = ['Gender', 'Company Type', 'WFH Setup Available']
stress_data = binarize_columns(stress_data, binary_cols)

# Aufteilung in X und y
y = stress_data['Burn Rate']
X = stress_data.drop('Burn Rate', axis=1)

# Train-Test-Split
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, shuffle=True, random_state=1)

# Skalierung von X
scaler = StandardScaler()
X_train_scaled = pd.DataFrame(scaler.fit_transform(X_train), index=X_train.index, columns=X_train.columns)
X_test_scaled = pd.DataFrame(scaler.transform(X_test), index=X_test.index, columns=X_test.columns)
r2_scores = []

# Instanzierung und Anpassen des Modells
lm = LinearRegression().fit(X_train_scaled, y_train)

# Vorhersagen
lm_y_pred = lm.predict(X_test_scaled)

# Auswertung des linearen Regressionsmodells
lr_r2, lr_mae = evaluate_model(y_test, lm_y_pred)

# Anzeigen der Metriken für das lineare Regressionsmodell
print("Linear Regression R2: ", lr_r2)
print("Linear Regression MAE: ", lr_mae)
```

```
# Modelle für die Regression in einem Tupel
models = [
    ("Linear Regression", LinearRegression()),
    ("Linear Regression (L2 Regularization)", Ridge()),
    ("K-Nearest Neighbors", KNeighborsRegressor()),
    ("Neural Network", MLPRegressor()),
    ("Support Vector Machine (Linear Kernel)", LinearSVR(max_iter=10000, tol=1e-5, dual=False, loss='squared_epsi
    ("Support Vector Machine (RBF Kernel)", SVR()),
    ("Decision Tree", DecisionTreeRegressor()),
    ("Random Forest", RandomForestRegressor()),
    ("Gradient Boosting", GradientBoostingRegressor())
]

# Erstellen eines leeren Dictionary zum Speichern der Ergebnisse
results_dict = {'Model': [], 'R2 Score': []}

# Schleife für Modelltraining und Bewertung
for name, model in models:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    print(f"{name} - R2 Score: {r2:.4f}, MAE: {mae:.4f}")
    results_dict['Model'].append(name)
    results_dict['R2 Score'].append(r2)
```

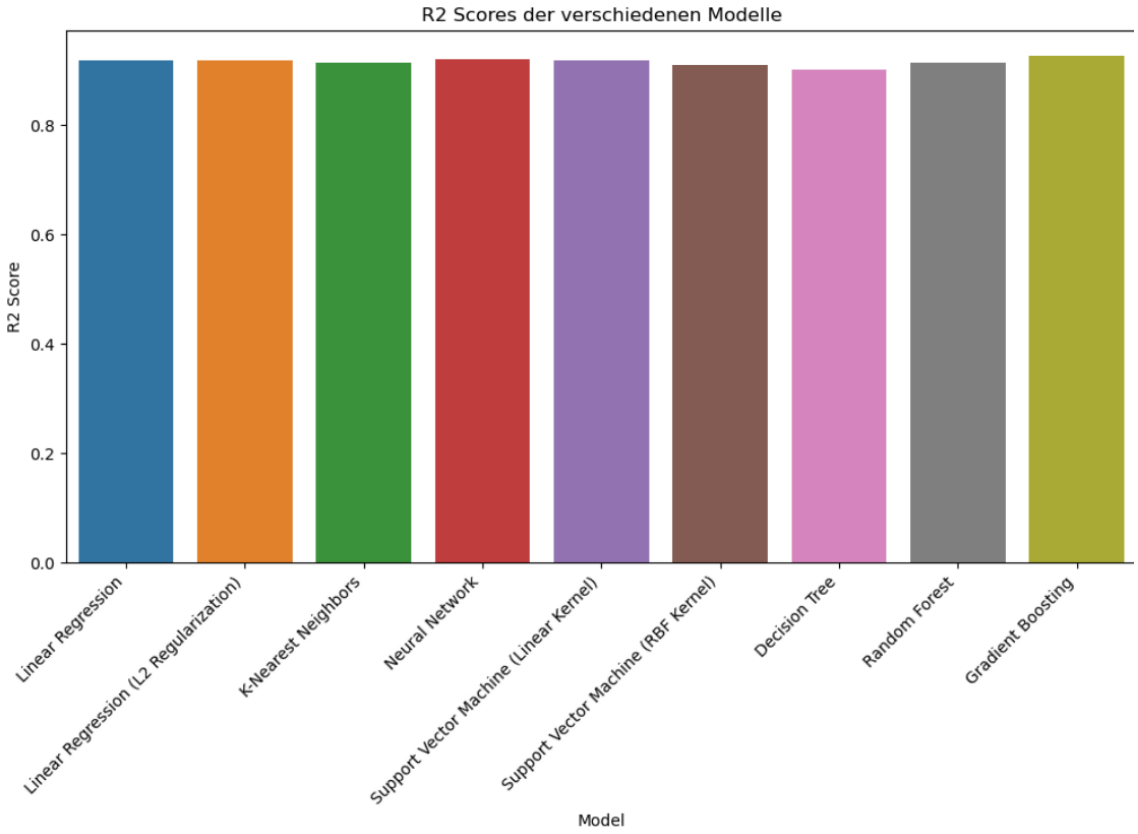
```
Linear Regression R2:  0.9188226742472478
Linear Regression MAE:  0.04595032032644779
Linear Regression - R2 Score: 0.9188, MAE: 0.0460
Linear Regression (L2 Regularization) - R2 Score: 0.9188, MAE: 0.0460
K-Nearest Neighbors - R2 Score: 0.9135, MAE: 0.0460
Neural Network - R2 Score: 0.9189, MAE: 0.0459
Support Vector Machine (Linear Kernel) - R2 Score: 0.9188, MAE: 0.0459
Support Vector Machine (RBF Kernel) - R2 Score: 0.9102, MAE: 0.0505
Decision Tree - R2 Score: 0.9005, MAE: 0.0484
Random Forest - R2 Score: 0.9135, MAE: 0.0458
Gradient Boosting - R2 Score: 0.9263, MAE: 0.0438
```

Alle Modelle haben sehr hohe R2 Scores und sehr niedrige MAE Werte. An den hohen R2 Scores lässt sich erkennen, dass die Variabilität sehr gut durch das Modell erklärt wird. Durch die niedrigen MAE Werte wird ersichtlich, dass der durchschnittliche absolute Fehler zwischen den tatsächlichen Werten und den vorhergesagten Werten des Modells sehr gering ist. Im Allgemeinen bedeuten ein hoher R2 Score und ein niedriger MAE Wert, dass das verwendete Modell aussagekräftig ist. Das Dictionary wird zur weiteren Verwendung in einen Data Frame umgewandelt:

```
# Konvertieren des Dictionary in ein DataFrame
results_df = pd.DataFrame(results_dict)
```

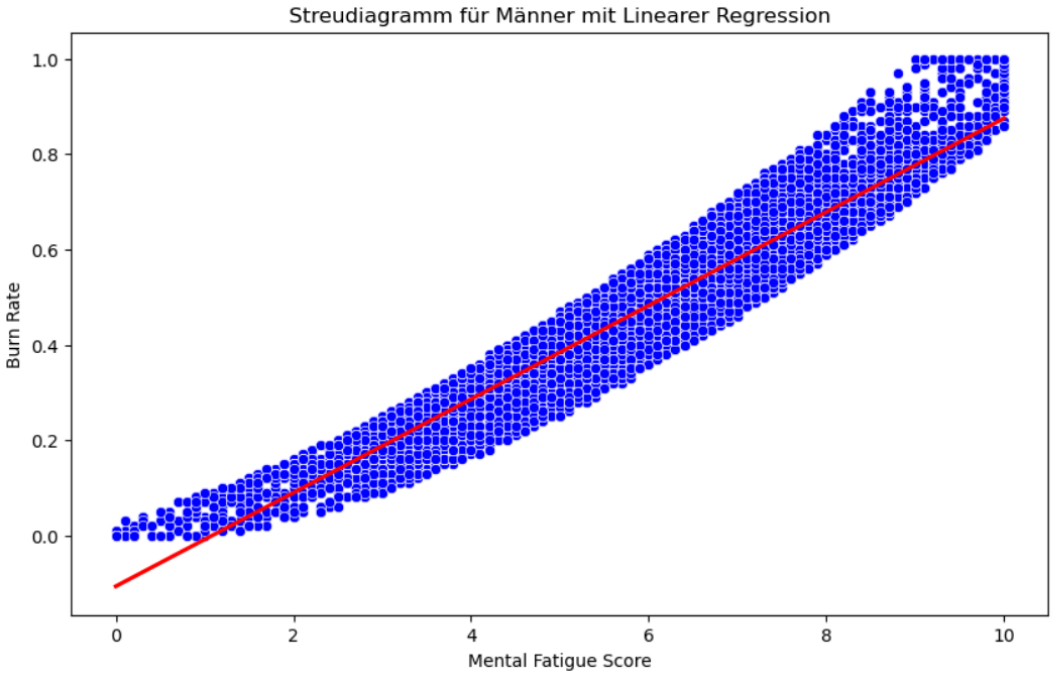
Zum Vergleich der einzelnen Modelle werden hier die verschiedenen R2 Scores visualisiert:

```
# Visualisierung der R2-Scores
plt.figure(figsize=(12, 6))
sns.barplot(x='Model', y='R2 Score', data=results_df)
plt.title('R2 Scores der verschiedenen Modelle')
plt.ylabel('R2 Score')
plt.xticks(rotation=45, ha='right')
plt.show()
```



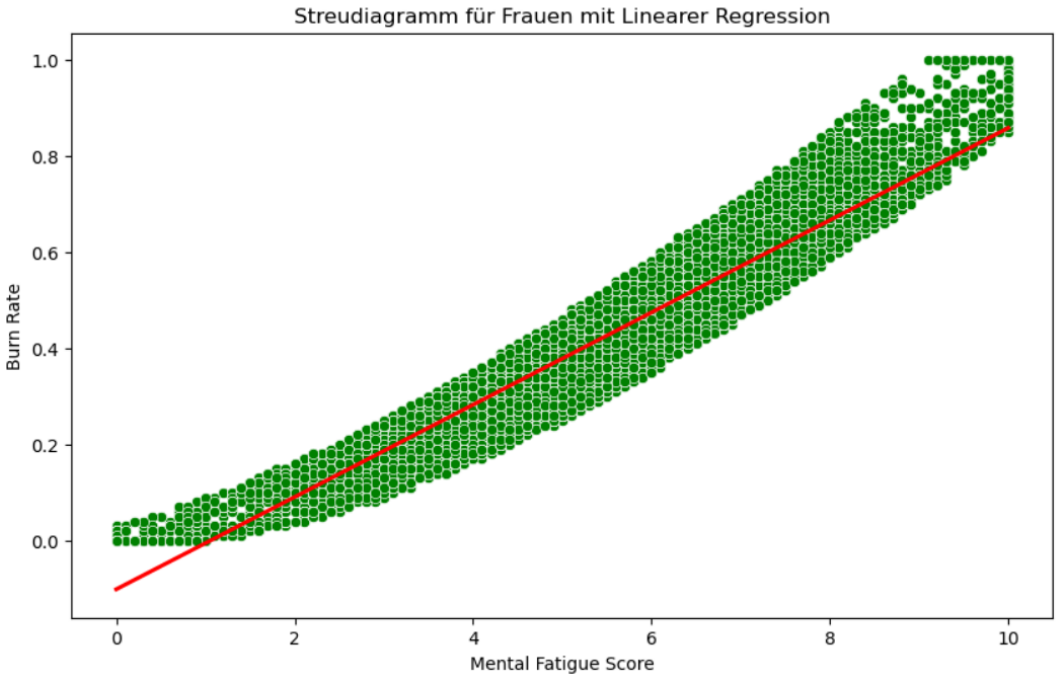
Es lässt sich nun auch graphisch erkennen, dass alle Modelle ähnlich hohe R2 Scores haben.
Visualisierung des linearen Regressionsmodells für Männer in Form eines Streudiagramms:

```
# Visualisierung für Männer
visualize_scatter_with_regression(male_data, 'Mental Fatigue Score', 'Burn Rate', 'blue',
'Streudiagramm für Männer mit Linearer Regression')
```



Visualisierung des linearen Regressionsmodells für Frauen in Form eines Streudiagramms:

```
# Visualisierung für Frauen
visualize_scatter_with_regression(female_data, 'Mental Fatigue Score', 'Burn Rate', 'green',
'Streudiagramm für Frauen mit Linearer Regression')
```



Aus den beiden obigen Graphiken wird deutlich, dass die lineare Regression bei Männern und Frauen annähernd identisch ist. Da beide Regressionen die Aussagekraft der Daten bestätigen, lässt sich schlussfolgern, dass sowohl bei Männern, als auch bei Frauen der Mental Fatigue Score mit der Burn Rate korreliert.

5 Fazit

Zusammenfassend lässt sich sagen, dass Burn Rate und Mental Fatigue Score stark miteinander korrelieren und ein WFH Setup einen positiven Einfluss auf den Mental Fatigue Score und somit auch auf die Burn Rate hat. Die Frage, ob dazwischen tatsächlich ein kausaler Zusammenhang besteht, bedarf allerdings weiterer Untersuchung.
Zudem lässt sich anhand des Datensatzes keine fundierte Aussage zum Zusammenhang zwischen Dauer der Unternehmenszugehörigkeit und Burn Rate treffen. Zur genaueren Betrachtung dieser Frage müsste ein weiterer Datensatz hinzugezogen werden, bei dem die Unterschiede in der Länge der Betriebszugehörigkeit stärker variieren.
Außerdem besteht der Verdacht, dass die Daten aus dem vorliegenden Datensatz eventuell computergeneriert sein könnten und nicht auf echten Werten basieren, da keine Informationen zur Datenerhebung vorliegen und nahezu keine Ausreißer vorhanden sind. Daher verliert die Interpretation unserer Ergebnisse ebenfalls an Aussagekraft. Hinzu kommt, dass der Datensatz wenige interessante Fragestellungen ermöglicht, die einer Analyse bedürfen. Mit der Erfahrung, die bei der Bearbeitung dieses Projekts gesammelt wurde, wäre die Entscheidung demnach auf einen anderen Datensatz gefallen.