Report

Gruppe D: Nico Gerspach & Jonas Lüttmann

1 Introduction and data

Nachdem wir uns einen Überblick über die möglichen Datenquellen verschafft haben, sind wir zum Entschluss gekommen, uns die Zensus Daten von 2011 genauer anzuschauen. Die Zensus Daten 2011 enthalten unter anderem Angaben über die Erwerbstätigkeit in Deutschland sowie unterschiedlichste soziodemografische Informationen (vgl. Zensus 2011). Hinsichtlich der Arbeitslosigkeit gibt es diverse Vorurteile und Vermutungen. So wird häufig behauptet, dass der Großteil der Arbeitslosen Personen mit Migrationshintergrund sind oder ein schlechtes Bildungsniveau vorweisen. Die Bundeszentrale für politische Bildung veröffentlichte hierzu einen Bericht, welcher diese zwei Vermutungen sogar bestätigt (vgl. Arbeitslosenquoten nach Geschlecht und Staatsangehoerigkeit, 2021), (vgl. Arbeitslosenquoten nach Bildung und Alter, 2021). Diese Informationen waren für uns Anreiz genug, um diese Zusammenhänge zu untersuchen.
Die Fragestellung, welche wir innerhalb dieses Projekts untersuchen, lautet wie folgt:

> **i** Fragestellung
>
> Gibt es einen Zusammenhang zwischen den soziodemografischen Merkmalen und der Arbeitslosenrate?

Die Arbeitslosigkeit im Allgemeinen ist ein Indikator für die Situation auf dem Arbeitsmarkt. Die Arbeitslosenquote kann unter Angabe der "Anzahl Erwerbslosen" sowie der "Anzahl Erwerbstätigen" berechnet werden. Als soziodemografische Merkmale haben wir uns für folgende sechs entschieden und wie folgt definiert (siehe Kapitel 5.1). Der Grund wieso wir uns für die 6 Merkmale entschieden haben war, da aufgrund der Zensus Umfrage die soziodemografischen Merkmale in verschiedene Kategorien aufgeteilt waren. Dies sind Informationen auf Gemeindeebene, wie z.B. Angaben zum Familienstand (Anzahl Personen die ledig sind, verheiratet, usw.), Angaben nach Relegionszugehörigkeit (Anzahl Personen die römisch-katholisch, evangelisch oder sonstiges) oder Angaben zum Bildungsniveau (Anzahl Personen ohne beruflischen Abschluss, mindestens eine Lehre, mindestens Hochschulabschluss, usw.). Aufgrund dieser Kategorien entschieden wir uns Quoten zu bilden, die unserer Meinung nach in einer Beziehung zur Arbeitslosenquote stehen können.

1.1 Import relevanter Module

1.2 Import Datensatz

1.3 Data Structure

Um einen ersten groben Überblick zu bekommen, geben wir uns eine Info über unseren Datensatz mit der Pandas-Funktion `pd.info()` aus. Hierbei ist zu sehen, dass 166 Spalten als Datentyp `Objekt` haben und somit auf gemischte Datentypen hindeutet (siehe Kapitel 5.2). In diesem ist ersichtlich, dass manche Spalten einige leere Zellen enthalten. So gibt es einige Spalten mit nur 2187 non-null Werten. Die ersten 10 Zeilen sowie die letzten 10 Zeilen des Datensatz verdeutlichen ebenso, dass nicht alle Werte nutzbar sind (siehe **?@tbl-erste10** & **?@tbl-letzte10**). Daher führen wir im nächsten Schritt Daten Korrekturen durch, um saubere Datentypen zu haben.

1.4 Data Correction

- `integer` in `float` umwandeln
- / und - in 0-Werte verwandeln, da diese im engeren Sinne als 0 zählen
- Zahlen in Klammern als normale Zahlen umwandeln

Anschließend erfolgt der Check, ob die Anpassung der Zahlen, auf Basis zweier bekannter Gemeinden mit ursprünglich nicht korrekt formatierten Werten, nun korrekt ist:

Die definierten predictor variables (siehe Kapitel 5.1) müssen im Folgenden noch berechnet werden:

1.4.1 Data splitting 1

Unter Angabe des Spaltenindex filtern wir den Dataframe, sodass wir nur noch die relevanten Spalten erhalten und kopieren diese Werte in ein neues Dataframe `df_analyse`:

Nicht jede Zelle in unserem Dataframe `df_analyse` enthält numerische Werte, da öfters die Info NaN angezeigt wird.

> Gründe, weshalb NaN vorkommt können folgende sein:
>
> - Daten sind nicht erhoben worden
> - Zahlenwert der Erfassung nicht sicher genug
> - Aufgrund von Geheimhaltungsverfahren

Die erneute Ausführung des Codes `pd.info()` zeigt, dass nur noch die relevanten Quoten enthalten sind und informative Spalten. Dabei ist ersichtlich, dass manche Spalten deutlich weniger NaN Werte enthalten (siehe Kapitel 5.3)

Da mit NaN Werten nicht gerechnet werden kann müssen diese Zeilen entfernt werden. Mithilfe der `.dropna` Funktion entfernen wir diese Zeilen deren relevanten Spalten NaN Werte enthalten. Im gleichen Schritt wird der Dataframe auf Hierarchie-Ebene **Gemeinde** gefiltert. Das neue Dataframe heißt nun `df_analyse_gemeinde` (Siehe **?@tbl-gemeinde**).

Die Ausführung der Funktion `pd.info()` ergibt folgende Übersicht. Unser nun bereinigter Dataframe enthält 1574 Zeilen, welcher die Basis für die Anwendung der Modelle bildet.

1.4.2 Variable List

Im Folgenden werden die Prädikatoren sowie die Outcome-Variable definiert. Zudem werden die Daten für die Prädikatoren sowie die Outcome-Variable definiert.

1.4.3 Data Splitting 2

Für das spätere Modell möchten wir Trainings- und Testdaten. Um die deskriptive und explorative Datenanalyse bereits auf den Trainingsdaten durchzuführen, splitten wir im nächsten Schritt die Daten. Dies machen wir mit der `train_test_split`-Funktion von scikit-learn.

1.5 Descriptive Analytics

Die statistischen Werte, die mithilfer der Funktion `.describe` ausgegeben werden, geben ein erstes Gefühl für die bereingten Daten, welche für die Erstellung der Modelle verwendet werden.

Die Funktion `.describe` enthält:

- Lagemaße (Mittelwert, Median)
- Streuungsmaße (Standardabweichung, Quartile)

1.6 Explorative Analytics

Um die Verteilung der zugrundeliegenden Daten grafisch darstellen zu können, erstellen für für jede Variable ein Histogramm. Das Histogramm für die "Christenquote" weist eine linksschiefe, multimodale Verteilung auf. Die "Männerquote" weist eine annähernd symetrische, unimodale Verteilung auf. Alle weiteren Variablen sind rechtsschief, unimodal verteilt.

Zur Visualisierung der Beziehungen zwischen Response- und Predictor Variables haben wir uns für die Anwendung von Streudiagrammen bzw. Scatter Plots entschieden. Jeden Prädikator haben wir mit der Arbeitslosenquote gegegnübergestellt, um erste Erkenntnisse gewinnen zu können. Jedoch ist es anhand der Scatter Plots zunächst schwierig zu erkennen, welcher Prädikator die höchste Korrelation mit der Arbeitslosigkeit aufweist, da die berechneten Variablen unterschiedliche Werte bzw. Verteilungen auf der X-Achse annehmen und kein direkter Vergleich stattfinden kann.

Zu erkennen ist, dass lediglich die Variable "Christenquote" eine moderate Korrelation mit der Arbeitslosenquote aufweist. Zwei weitere Quoten, die noch eine leichte bis moderarte Korrelation haben sind die "Singlequote" sowie die "Migrationsquote".

Pearson Korrelations-Koeffizienten

Unter Anwendung der Funktion `.corr` werden die Korrelationen berechnet und in Tabellenform ausgegeben.

Die Pearson Korrelations-Koeffizienten bestätigen die oben genannten Vermutungen.

Korrelationen

- Moderate Korrelation mit der Arbeitslosenquote: *"Christenquote"*
- Leichte bis moderate Korrelation: *"Singlequote"*
- Äußerst geringe Korrelation: *"Akademikerquote"*

2 Methodology

Wie in der Introduction beschrieben haben wir uns aufgrund der Literatur-Recherche wie auch nach Betrachtung des Data Sets für 6 Quoten entschieden, welche die Arbeitslosenquote beeinflussen könnten.

Auf Basis der explorativen Analyse wurde bereits an dieser Stelle beschlossen, nur die "Christenquote", "Singlequote" und die "Migrationsquote" in die Auswahl für die Modelle aufzunehmen. Da in den Scatterplots linere Zusammenhänge erkennbar sind, wollen wir versuchen, diese mit lineran Modellen zu erklären.

Zu Beginn der jeweiligen Modellierungsprozesse haben wir jeweils das Regressionsmodell ausgewählt. Der nächste Schritt war es die Modelle zu validieren und an die Daten anzupassen. Die Validierung haben wir mit der Cross-Validation umgesetzt und dabei den Mean Squared error pro Fold angeschaut.

Daraufhin haben wir mit der Funktion `reg.fit` die Modelle an die Traingsdaten angepasst und somit trainiert und den Intercept mit der y-Achse sowie den/die Koeffizienten berechnet.

Der abschließende Schritt war die Evaluierung mit den Test-Daten. Schlussendlich haben wir die Modelle anhand verschiedener Gütemaße (R^2, MSE, RMSE, MAE) bewertet.

Durch die Peer-Review haben wir das Feedback bekommen, dass wir für die Multiple Regression die Stepwise Selection nutzen könnten, was wir dann im Folgenden auch umgesetzt haben.

Diese grob beschriebenen Schritte wollen wir im Folgenden detaillierter erklären.

2.1 Lineare Regression

2.1.1 Modell-Auswahl

Zunächst werden für die einzelnen Quoten drei Modelle mit jeweils der Linearen Regression definiert. Hier greifen wir auch auf scikit-learn zurück und nehmen `LinearRegression` als Modell.

2.1.2 Training & Validation

Im nächsten Schritt werden die Modelle trainiert und validiert. Dies wird mit der Cross-Validation durchgeführt. Hierbei berechnen wir für jedes Modell den Mean-Squared-Error für je fünf Folds. Hierfür nehmen wir die Funktion `cross_val_score` und visualisieren das zum einen in einer Tabelle und zum anderen als Liniendiagramm. Dabei sehen wir, dass die Mean-Squared-Erros je Fold voneinander abweichen. Grundsätzlich ist der MSE für die "Christenqutote" aber immer am geringsten, weshalb wir uns im Rahmen der Linearen Regression auf die "Christenquote" konzentrieren.

2.1.3 Fit Model

Im nächsten Schritt möchten wir das Modell an die Trainingsdaten anpassen und den y-Achsenabschnitt sowie die Steigung berechnen. Hierfür nutzen wir die `.fit` Funktion.

Dabei ergeben sich für das lineare Modell mit der "Christenquote" als predictor folgende Werte:

2.1.4 Evaluation on test set

Im nächsten Schritt evaulieren wir unser Modell mit den Testdaten. Dazu prognostizieren wir y-Werte. Das bedeutet "Arbeitslosenquoten" auf Basis der Testdaten, welche diverse "Christenquoten" darstellen. Dies setzen wir mit der Funktion `.predict` um.

Für die prognostizierten Werte berechnen wir Gütemaße, wie den R-squared (R2), den Mean-Squared-Error (MSE), den Rooted-Mean-Squared-Error (RMSE) sowie den Mean-Absolute-Error (MAE).

2.2 Multiple Regression

2.2.1 Modell-Auswahl

Auch für die Multiple Regression nehmen wir von scikit-learn die `LinearRegression` als Modell. Der Unterschied zum vorherigen Abschnitt ist allerdings, dass wir hierbei mehr als eine predictor-Variable berücksichtigen.

2.2.2 Training & Validation

Zuerst nehmen wir alle sechs predictor-Variablen, um das Modell der Multiplen Regression zu trainieren und validieren. Hierfür nutzen wir den gleichen Cross-Validation Ansatz wie bei der Linearen Regression.

Im nächsten Schritt möchten wir die Wrapper-Methoden nutzen, um zu prüfen, ob die Multiple Regression mit weniger Features besser performt und somit leichter verständlich wird. Hierfür nutzen wir von scikit-learn den `SequentialFeatureSelector`. Nachdem wir die Anzahl der ausgewählten Features selektiert haben und den MSE für je fünf Folds angeschaut haben, kamen wir zu der Erkenntnis, dass die Multiple Regression mit der Anzahl von fünf Features am besten performt. Diese Variationen haben wir mit der Forward- und Backward-Selection durchgeführt.

Das Ergebnis der Backward- und Forward-Selection sind jeweils die gleichen fünf Features. Das ist eher untypisch. Obwohl der `SequentialFeatureSelector` bereits eine Cross-Validation mit fünf Folds durchführt, machen wir das für auch für die fünf selektierten Variablen, um die beiden Multiplen Regressionen zu vergleichen.

> **Erkenntnis**
>
> Hierbei sehen wir, dass die Mean-Squared-Erros für die Multiple Regression mit fünf Features niedriger sind, weshalb wir im weiteren Verlauf diese fünf Features für die Multiple Regression berücksichtigen werden.

2.2.3 Fit Model

In diesem Schritt passen wir wieder das Modell an die Trainingsdaten an und berechnen den y-Achsenabschnitt sowie die Steigung.

2.2.4 Evaluation on test set

Im nächsten Schritt evaulieren wir unser Modell mit den Testdaten. Dazu prognostizieren wir y-Werte. Das bedeutet "Arbeitslosenquoten" auf Basis der Testdaten, welche diverse "Christenquoten" darstellen. Dies setzen wir erneut mit der Funktion `.predict` um.

Für die prognostizierten Werte berechnen wir Gütemaße, wie den R-squared (R2), den Mean-Squared-Error (MSE), den Rooted-Mean-Squared-Error (RMSE) sowie den Mean-Absolute-Error (MAE).

2.3 Lasso Regression

2.3.1 Modell-Auswahl

Bei der Lasso Regression ist es zuerst notwendig, die Variablen zu standardisieren, da Lasso am besten performt, wenn die Features um den Wert 0 zentriert sind. Für diese Standardisierung definieren wir zuerst mit der `StandardScaler().fit`-Funktion unseren Skalierer. Diese scikit-learn Funktion entfernt den Mittelwert und skaliert jedes Feature auf eine Einheitsvarianz. Das wird für jedes Feature getrennt durchgeführt. Im nächsten Schritt transformieren wir unsere Trainings- und Testdaten mit diesem Skalierer. Für die Berechnung des alpha-Wertes für Lasso Regression nehmen wir von scikit-learn die `LassoCV`-Funktion, welche bereits eine Cross-Validation integriert hat.

2.3.2 Training & best alpha

Die `LassoCV`-Funktion hat wie bereits erwähnt, die Cross Validation integriert. Somit ermittelt das Model für uns den bestmöglichen alpha-Wert, um die Regression durchzuführen. Nachdem wir unser Modell trainiert haben, können wir uns mit `.alpha_` den bestmöglichen alpha-Wert ausgeben lassen:

2.3.3 Fit Model

Anschließend nutzen wir den besten Wert für alpha, um die Lasso Regression damit durchzuführen. Mit dem besten Wert für alpha, werden folgende Koeffizienten berechnet:

2.3.4 Evaluation on test set

Wie in den vorherigen beiden Modellen evaluieren wir unser Modell mit den Test-Daten und prognostizieren Werte.

Für die prognostizierten Werte berechnen wir Gütemaße, wie den R-squared (R2), den Mean-Squared-Error (MSE), den Rooted-Mean-Squared-Error (RMSE) sowie den Mean-Absolute-Error (MAE).

3 Results

Mithilfe der durchgeführten Analyse konnte festgestellt werden, dass die stärkste positive Korrelation (nach Pearson) zwischen der Arbeitslosenquote einer Gemeinde und dem Prädikator “Singlequote” besteht (r = +0.426). Die stärkste negative Korrelation besteht zwischen der Arbeitslosenquote einer Gemeinde und dem Prädikator “Christenquote” (r = -0.6fünf3). Die schwächste Korrelation, auf Basis unserer Daten, weist die “Akademikerquote” auf (r=-0.121).

Das Ergebnis der drei durchgeführten Modelle lautet wie folgt:

Betrachtet man die Ergebnisse in der Tabelle, so fällt auf, dass die Multiple Regression bei fast allen Statistiken am besten abschneidet. Sie hat zum einen den den größten R2-Wert, den niedrigsten Mean Squared Error, dadurch folglich auch den niedrigsten Rooted Mean Squared Error. Allein der Mean Absolute Error ist minimal höher als bei der Lasso Regression. Zudem fällt auf, dass die Lasso und Multiple Regression in diesen Statistiken eigentlich genau gleich sind. Der Grund dafür ist, dass der gewählte alpha-Wert sehr nahe an null ist und somit so gut wie keinen Einfluss hat. Zudem berücksichtigt zwar die Lasso-Regression alle predictor-Variablen, allerdings beträgt die Steigung der “Migrationsquote” 0, was bedeutet, dass diese keinen Einfluss hat.

Letztendlich haben wir uns für die Multiple Regression als bestes Modell entschieden, um die Abhängigkeit zwischen den soziodemografischen Merkmalen und der Arbeitslosenquote zu beschreiben. Daher speichern wird das Modell und stellen es im Ordner `../models/` bereit.

Unser Modell beinhaltet alle predictor-Variablen, außer der “Migrationsquote”. Diese ist im Zuge der schrittweisen Selektion herausgefallen, da mit ihr das Modell schlechter performt hätte. Der r-squared Wert bedeutet, dass unser Modell 62,5 % der Variabilität in der Arbeitslosenquote abdeckt. Der RMSE zeigt uns, dass die prognostizierte Arbeitslosenquote im Mittel um 1.297% um den tatsächlichen Wert streut. Zudem ist im Mittel der absolute Fehler auch bei 0.974%. Stellt man diese Werte ins Verhältnis zu der Spannweite der Arbeitslosenquote, so sind das Werte, die ins Gewicht fallen. So kann eine prognostizierte Arbeitslosenquote in einer Spannweite von beispielsweise 4%-ca.6% liegen.

Bei der Betrachtung der Koeffizienten der predictor-Variblen, fällt auf, dass die “Christenquote” mit nur -0.065 einen sehr niedrigen Wert hat, obwohl sie am Stärksten mit der Arbeitslosenquote korreliert. Die “Männerquote”, die nicht so stark mit der Arbeitslosenquote korreliert, hat hingegen mit -0.427 einen deutlichen stärkeren Einfluss auf das Modell. Nimmt man an, dass alle predictor-Variablen konstant sind und nur die “Männerquote” um eine Einheit steigt, so würde die die Arbeitslosenquote um 0.427% abnehmen. Diese Erkenntnisse in Verbindung mit dem Wissen, dass die Koeffizienten sich bei der Hinzunahme der “Migrationsquote” stark verändern, deutet auf eine starke Multikollinearität zwischen den predictor-Variablen hin. Auffällig ist auch, dass die Korrelation zwischen “Christenquote” und “Migrationsquote” bei ca. 0.41 liegt und damit ein weiterer Hinweis auf Multikollinearität ist. Das beeinträchtigt die Qualität unseres Modells. Dies lässt sich auch schon aus der Korrelationsübersicht erkennen, da hier einige predictor-Variablen untereinander stärken korrelieren als mit der Arbeitslosenquote selbst.

Zusammenfassend lässt sich sagen, dass unser Modell zwar mehr als die Hälfte der Variablität in der Arbeitslosenquote erklären kann, allerdings trotzdem nicht aussagekräftig genug ist, um die Auswirkungen der soziodemografischen Merkmale auf die Arbeitslosenquote zu erklären. Dies hängt vor allem auch mit der Multikollinearität der Variablen zusammen.

Folgende Herausforderungen gab es bei der Durchführung des Projekts:

> **i** Herausforderungen:
>
> 1. Großer Dataframe, obwohl nur wenige Daten tatsächlich relevant waren.
> 2. Ein Großteil der Zellen relevanter Spalten war leer, da durch die Zensus Umfrage die Werte entweder nicht ermittelt werden konnten oder zu ungenau waren.

3. Geringe Korrelation zwischen Arbeitslosenquote und den Variablen.

4 Discussion + Conclusion

Grundsätzlich macht es Sinn, dass mehrere soziodemografische Merkmale bei der Anwendung der Modelle berücksichtigt werden, da diese Modelle dann besser performen. Dies war bereits aus der Korrelationsübersicht erkennbar, wie auch später bei der Methodology, als verschiedene Modelle untersucht worden sind.

Aufgrund der durchgeführten Data Corrections wurden die tatsächlich genutzten Zeilen bzw. Gemeinden stark reduziert. Von insgesamt 11.339 Gemeinden wurden schlussendlich nur 1.574 Gemeinden berücksichtigt. Daher wäre es spannend zu wissen, ob die berücksichtigten Gemeinden repräsentativ für ganz Deutschland sind. Hierzu vergleichen wir den original Dataframe mit dem bereinigten Dataframe anhand der berücksichtigten Anzahl Einwohner sowie Anzahl Gemeinden.

4.1 Bereinigter DF

4.2 Original DF

Beim Vergleich der Anzahl Einwohner zwischen originalem DF und bereinigtem DF, fällt auf, dass in allen Gemeinden 80.029.997 Einwohner leben. Der bereinigte DF enthält jedoch nur 58.797.595 Einwohner. Die entspricht ca. 73.47 % der gesamten Einwohnerzahl Deutschlands. Die Summe aller Gemeinden beträgt 11.339 Stück. Bereinigt wurden jedoch nur 1.574 Gemeinden berücksichtigt, was einem Anteil von ca. 13.87 % entspricht.

Zusätzlich untersuchten wir auch noch, wie viele Gemeinden je Bundesland in unserem Projekt tatsächlich berücksichtigt worden sind. Auffällig ist, dass es Bundesländer gibt, bei denen lediglich ein Bruchteil der Gemeinden im bereinigten Dataframe enthalten ist. Z.B. Rheinland-Pfalz, Mecklenburg-Vorpommern, Schleswig-Holstein und Thüringen.

> **i** Erkenntnis
>
> Dies lässt darauf schliessen, dass in nur 13.87% der Gemeinden Deutschlands, 73.47% der Bevölkerung lebt. Somit wurden für die Untersuchung der Fragestellung überwiegend bevölkerungsreiche Gemeinden berücksichtigt, da kleine Gemeinden tendeziell weniger Angaben machten und somit herausgefiltert wurden. Dadurch, dass auch der Anteil der Gemeinden innerhalb der Bundesländer unterschiedlich hoch ausfällt, ist es fragwürdig, ob die berücksichtigten Daten das gesamte Land gut widerspiegeln.

Es ist in Zukunft hilfreich den Dataframe bereits bei Projektbeginn auf Vollständigkeit der relevanten Variablen zu prüfen. Sollten viele Werte fehlen, wäre es sinnvoll, dass Methoden angewendet werden, die fehlende Werte durch berechnete Werte ersetzen können.

Zudem sollte das Modell auch näher auf die Multikollinearität geprüft werden, da dies die Qualität des Modells stark beeinträchtigt.

5 Appendix

5.1 Appendix Predictor Variables

5.2 Appendix df_bevoelkerung

5.3 Appendix df_analyse

5.4 Appendix Dataframe bereinigt