CSE351 Final Project

Project#6: Covid-19 in Germany Analysis

Dayoung(Jennie) Yoon
Mohidul Abedin

Project Description

Problem: Covid cases in Germany

Coronavirus disease 2019 (COVID-19) was a contagious disease caused by severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). Even though, COVID has stabilized now compared to 2019, this disease has since spread worldwide, leading to an ongoing pandemic.

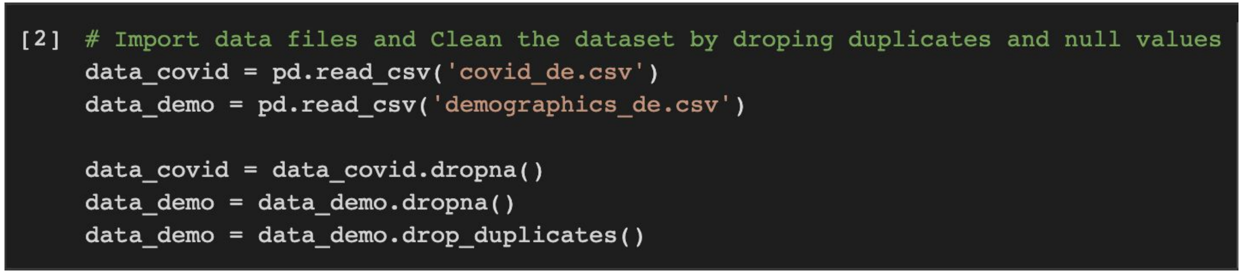
Datasets:
The project used Google Health COVID-19 Open Data Repository dataset.
https://github.com/GoogleCloudPlatform/covid-19-open-data
Our team referred to the demographic dataset of Germany and the Covid dataset which includes some columns such as age group, gender and death.

Analysis Step:
1. Exploratory Data Analysis
2. Modeling(ML Algorithms)
3. Evaluating

Data Cleaning

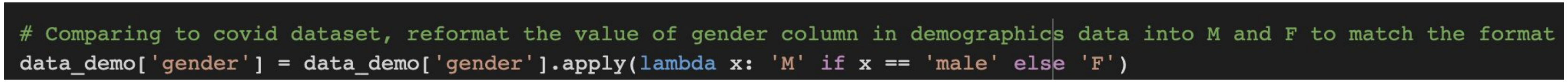
Before moving on to data merging and analysis, we cleaned the dataset by removing the outliers.

1. **Drop duplicates and null values**



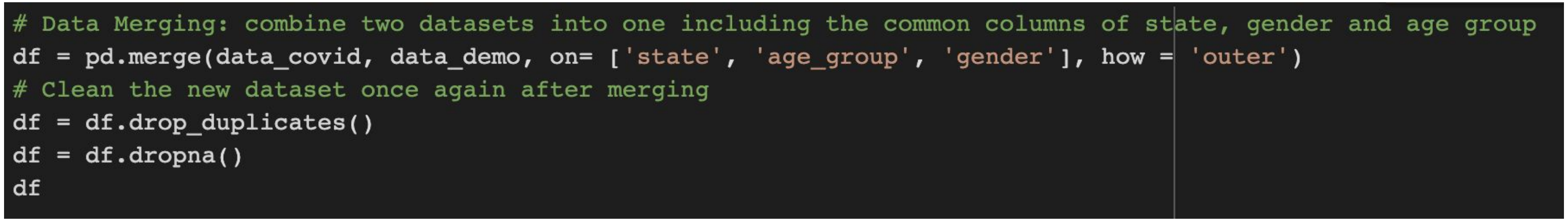
```
[2] # Import data files and Clean the dataset by droping duplicates and null values
    data_covid = pd.read_csv('covid_de.csv')
    data_demo = pd.read_csv('demographics_de.csv')

    data_covid = data_covid.dropna()
    data_demo = data_demo.dropna()
    data_demo = data_demo.drop_duplicates()
```

2. **Reformatting before combining the datasets**



```
# Comparing to covid dataset, reformat the value of gender column in demographics data into M and F to match the format
data_demo['gender'] = data_demo['gender'].apply(lambda x: 'M' if x == 'male' else 'F')
```

Data Merging



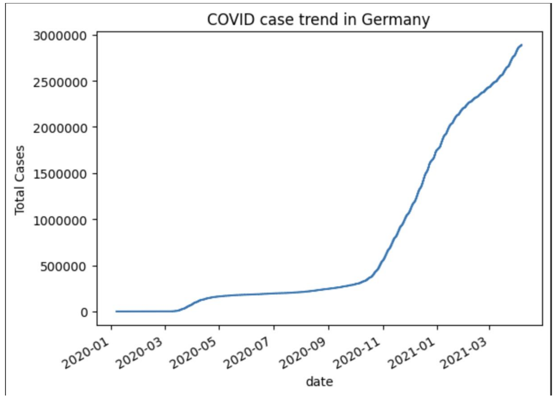
```
# Data Merging: combine two datasets into one including the common columns of state, gender and age group
df = pd.merge(data_covid, data_demo, on= ['state', 'age_group', 'gender'], how = 'outer')
# Clean the new dataset once again after merging
df = df.drop_duplicates()
df = df.dropna()
df
```

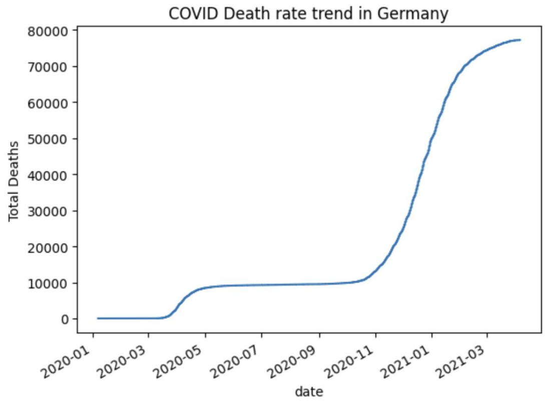
After cleaning each datasets, we merged the demographics and covid dataset into one and do the cleaning once again.

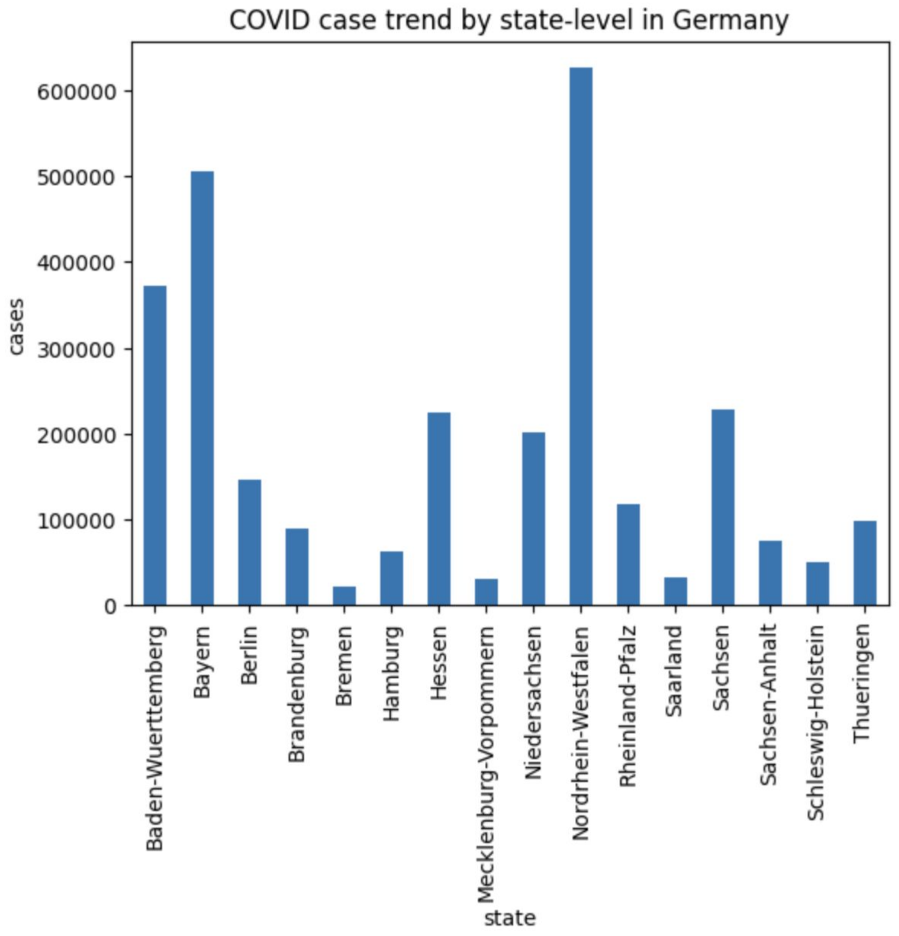
EDA (Exploratory Data Analysis)

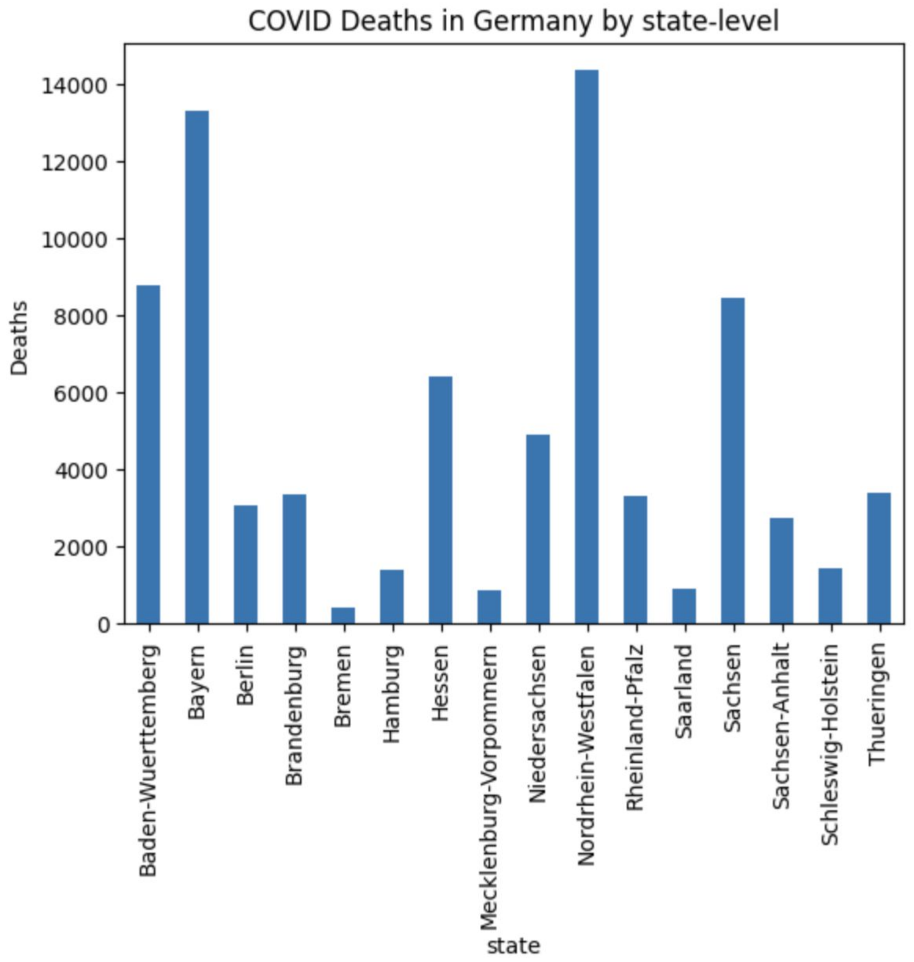
1. **What is the covid case trend in Germany, and how is it different from each state/county? Which state/county has the highest/lowest increasing rate?**
2. **What is the covid death rate trend in Germany, and how is it different from each state/county? Which state/county has the highest/lowest increasing rate?**
3. **Which age/gender group has the highest covid positive cases?**
4. **Which age/gender group has the highest covid death cases?**

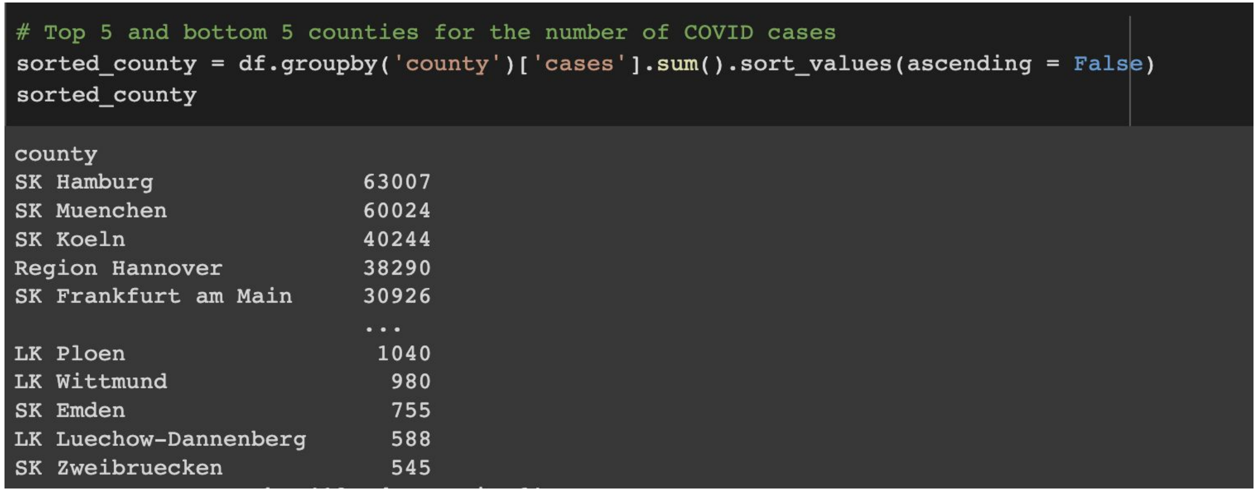
In order to get familiar with the dataset and decide what features and observations will be useful, we focused on the above questions and discovered the following.





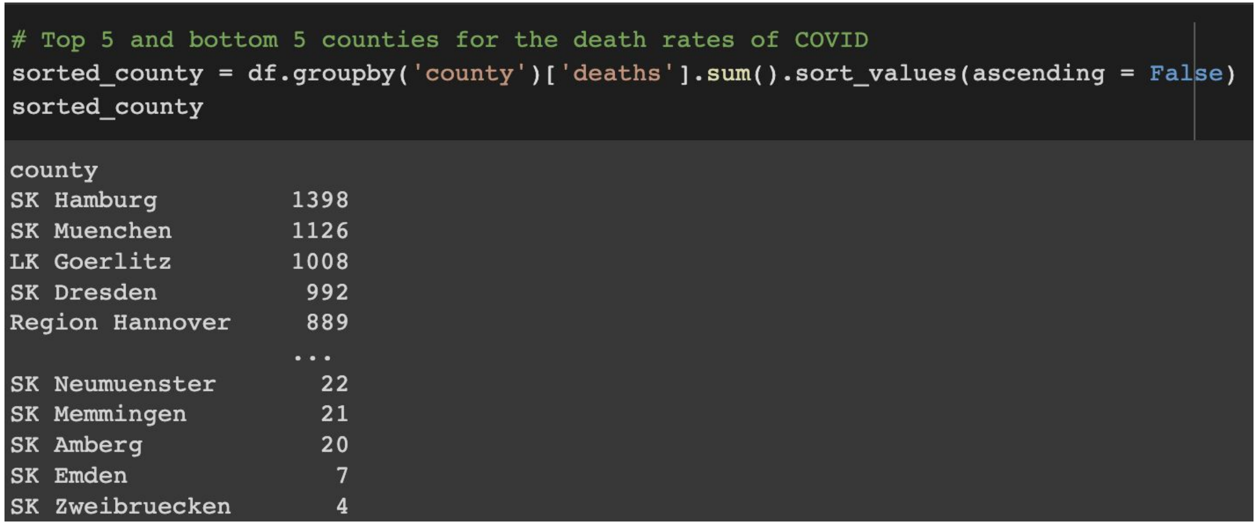

COVID case trend by state-level in Germany
cases
600000
500000
400000
300000
200000
100000
0
Baden-Wuerttemberg
Bayern
Berlin
Brandenburg
Bremen
Hamburg
Hessen
Mecklenburg-Vorpommern
Niedersachsen
Nordrhein-Westfalen
Rheinland-Pfalz
Saarland
Sachsen
Sachsen-Anhalt
Schleswig-Holstein
Thueringen
state


COVID Deaths in Germany by state-level
Deaths
14000
12000
10000
8000
6000
4000
2000
0
Baden-Wuerttemberg
Bayern
Berlin
Brandenburg
Bremen
Hamburg
Hessen
Mecklenburg-Vorpommern
Niedersachsen
Nordrhein-Westfalen
Rheinland-Pfalz
Saarland
Sachsen
Sachsen-Anhalt
Schleswig-Holstein
Thueringen
state



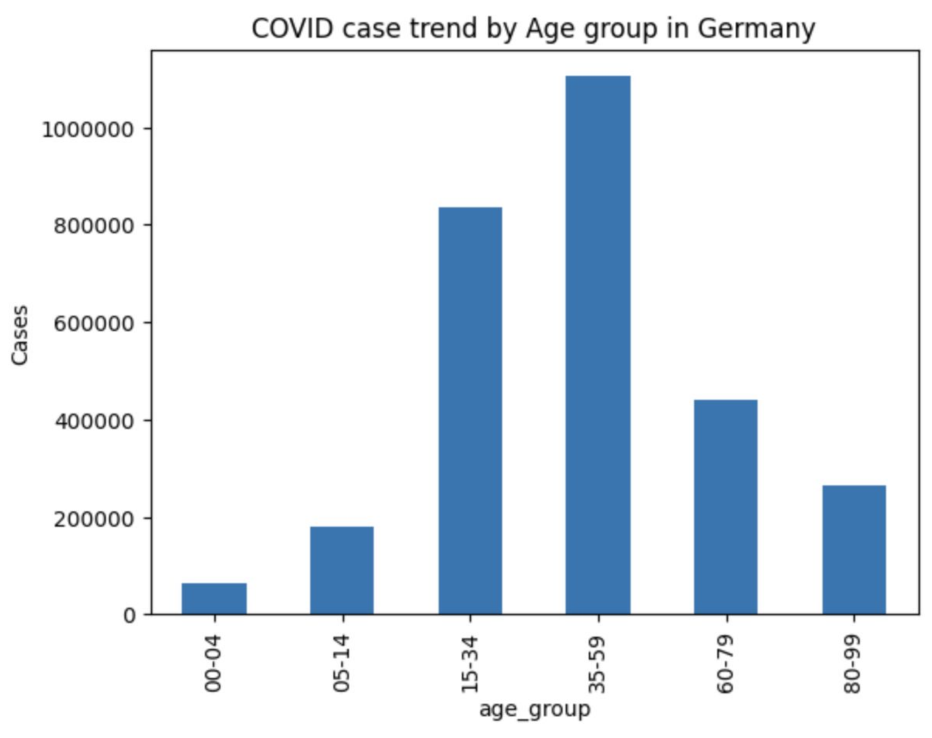
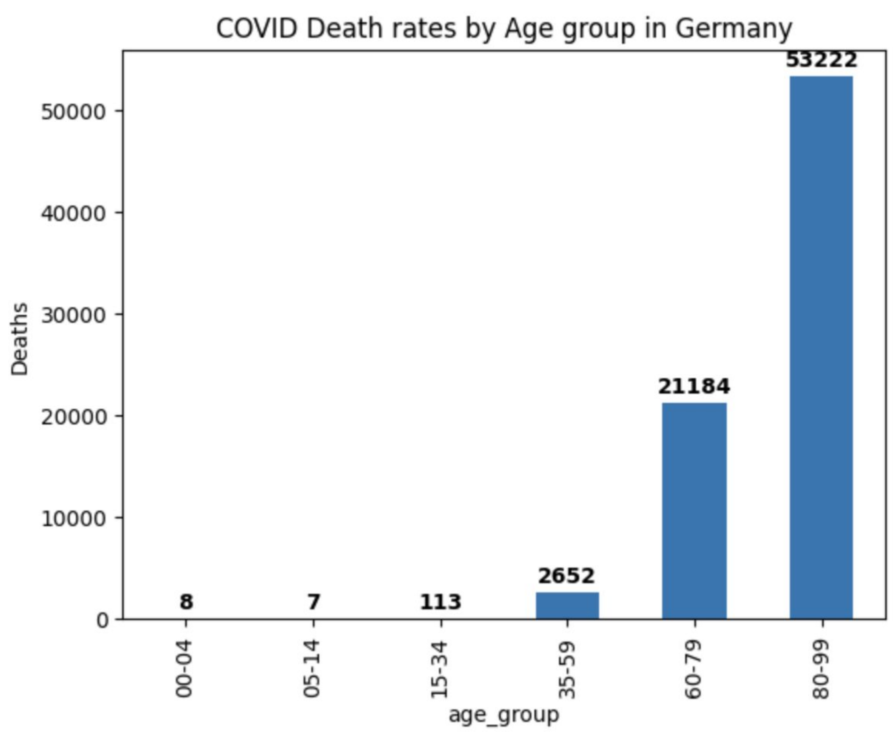
```
# Top 5 and bottom 5 counties for the number of COVID cases
sorted_county = df.groupby('county')['cases'].sum().sort_values(ascending = False)
sorted_county
```

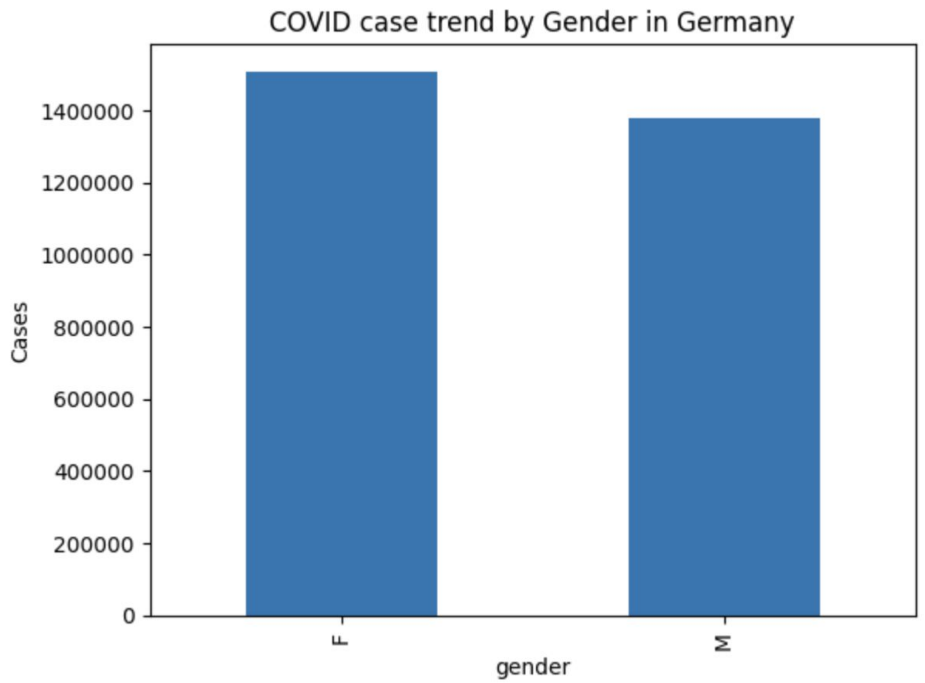
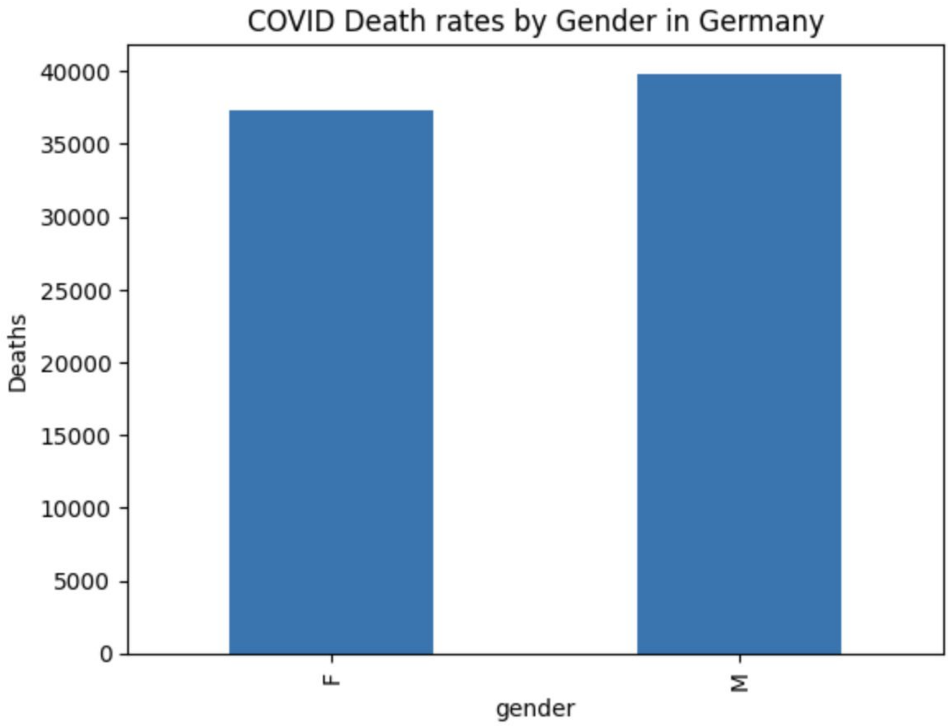
```
county
SK Hamburg                63007
SK Muenchen               60024
SK Koeln                  40244
Region Hannover           38290
SK Frankfurt am Main      30926
                           ...
LK Ploen                   1040
LK Wittmund                 980
SK Emden                    755
LK Luechow-Dannenberg       588
SK Zweibruecken             545
```



```
# Top 5 and bottom 5 counties for the death rates of COVID
sorted_county = df.groupby('county')['deaths'].sum().sort_values(ascending = False)
sorted_county
```

```
county
SK Hamburg          1398
SK Muenchen         1126
LK Goerlitz         1008
SK Dresden           992
Region Hannover      889
                     ...
SK Neumuenster        22
SK Memmingen          21
SK Amberg             20
SK Emden               7
SK Zweibruecken        4
```


COVID case trend by Age group in Germany
Cases
1000000
800000
600000
400000
200000
0
00-04
05-14
15-34
35-59
60-79
80-99
age_group

COVID Death rates by Age group in Germany
Deaths
50000
40000
30000
20000
10000
0
8
7
113
2652
21184
53222
00-04
05-14
15-34
35-59
60-79
80-99
age_group


COVID case trend by Gender in Germany
Cases
1400000
1200000
1000000
800000
600000
400000
200000
0
F
M
gender

COVID Death rates by Gender in Germany
Deaths
40000
35000
30000
25000
20000
15000
10000
5000
0
F
M
gender

Machine Learning Models

Logistic Regression

Binary classification

A statistical method used to model the probability of a binary outcome (such as survival or not) based on input features, like gender and age group in this case.

Random Forest

Ensemble classification

An ensemble learning method that constructs multiple decision trees during training and combines their predictions to improve classification accuracy and prevent overfitting.

Linear Regression

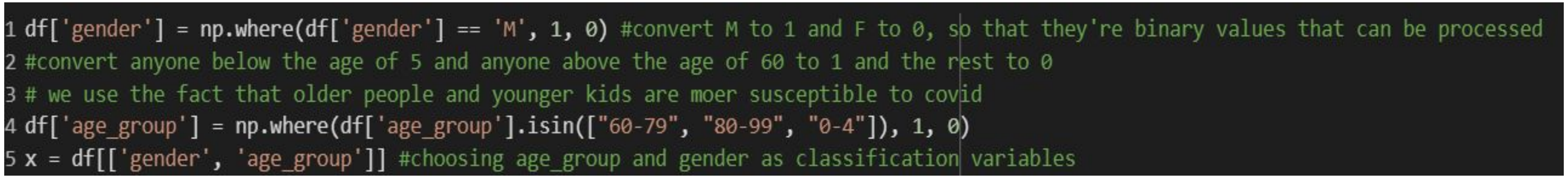
Continuous prediction

A linear approach to modeling the relationship between a dependent variable (recovered cases) and one or more independent variables (population), by fitting a straight line to the observed data points.

1. Logistic Regression

1. Preprocessing

In order to make a binary classification for logistic regression, we need to pick 2 features and turn them into binary quantities. We chose gender and age group.



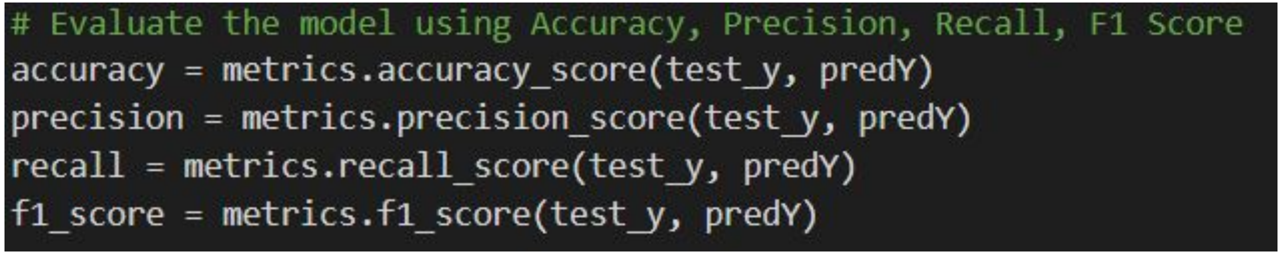
```python
df['gender'] = np.where(df['gender'] == 'M', 1, 0) #convert M to 1 and F to 0, so that they're binary values that can be processed
#convert anyone below the age of 5 and anyone above the age of 60 to 1 and the rest to 0
# we use the fact that older people and younger kids are moer susceptible to covid
df['age_group'] = np.where(df['age_group'].isin(["60-79", "80-99", "0-4"]), 1, 0)
x = df[['gender', 'age_group']] #choosing age_group and gender as classification variables
```

2. Prediction

We then separate the dataset into 70% Training and 30% Testing and run the logistic regression model.

3. Analysis

Then we evaluate the model using Accuracy, Precision, Recall, F1 Score

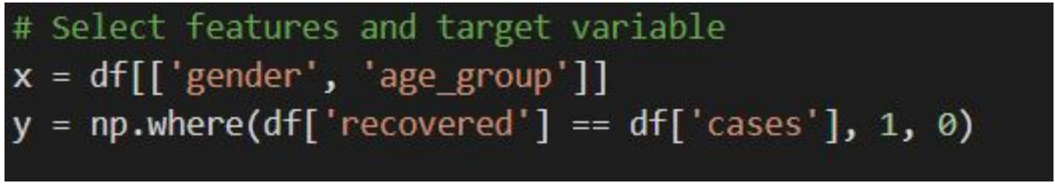


```python
# Evaluate the model using Accuracy, Precision, Recall, F1 Score
accuracy = metrics.accuracy_score(test_y, predY)
precision = metrics.precision_score(test_y, predY)
recall = metrics.recall_score(test_y, predY)
f1_score = metrics.f1_score(test_y, predY)
```

2. Random Forest Classifier

1. Preprocessing

We similarly used gender and age group again for the random forest classifier.



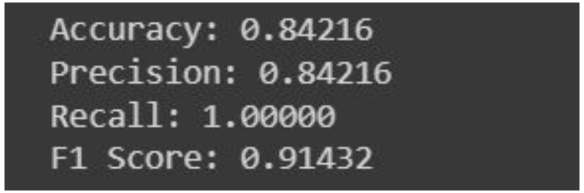
```
# Select features and target variable
x = df[['gender', 'age_group']]
y = np.where(df['recovered'] == df['cases'], 1, 0)
```

2. Prediction

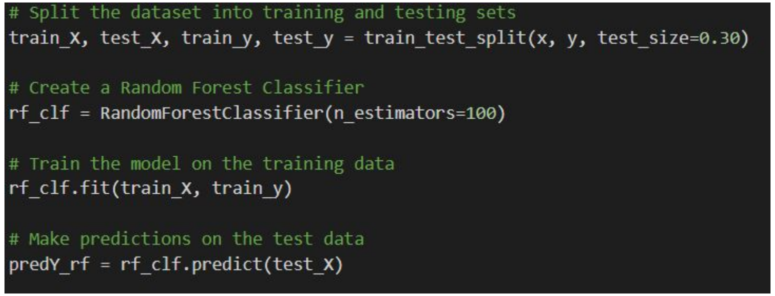
Again we separate the dataset into 70% Training and 30% Testing and run the random forest classifier model with n=100.

3. Analysis

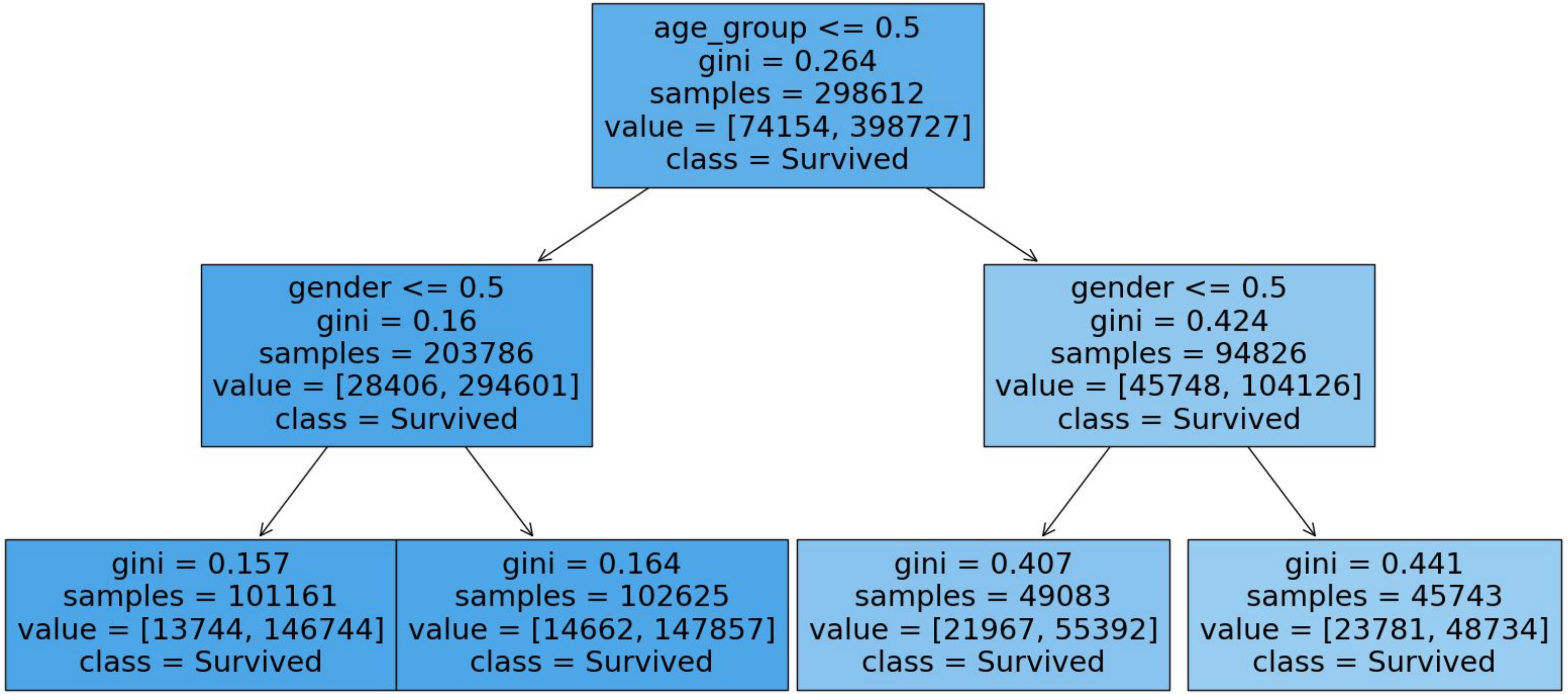
Then we evaluate the model using Accuracy, Precision, Recall, F1 Score



```
Accuracy: 0.84216
Precision: 0.84216
Recall: 1.00000
F1 Score: 0.91432
```



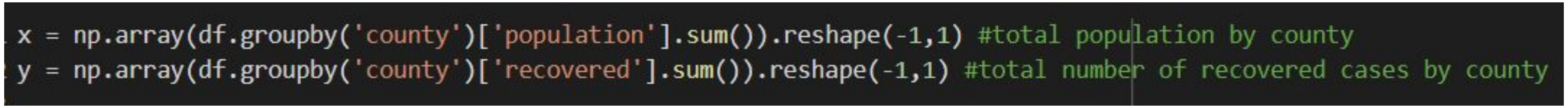
```
# Split the dataset into training and testing sets
train_X, test_X, train_y, test_y = train_test_split(x, y, test_size=0.30)

# Create a Random Forest Classifier
rf_clf = RandomForestClassifier(n_estimators=100)

# Train the model on the training data
rf_clf.fit(train_X, train_y)

# Make predictions on the test data
predY_rf = rf_clf.predict(test_X)
```


age_group <= 0.5
gini = 0.264
samples = 298612
value = [74154, 398727]
class = Survived
gender <= 0.5
gini = 0.16
samples = 203786
value = [28406, 294601]
class = Survived
gender <= 0.5
gini = 0.424
samples = 94826
value = [45748, 104126]
class = Survived
gini = 0.157
samples = 101161
value = [13744, 146744]
class = Survived
gini = 0.164
samples = 102625
value = [14662, 147857]
class = Survived
gini = 0.407
samples = 49083
value = [21967, 55392]
class = Survived
gini = 0.441
samples = 45743
value = [23781, 48734]
class = Survived

3. Linear Regression

1. Preprocessing

In order to do a linear regression, we sum up the population by county and the amount of people recovered by county to see if there is a relationship between the two variables.



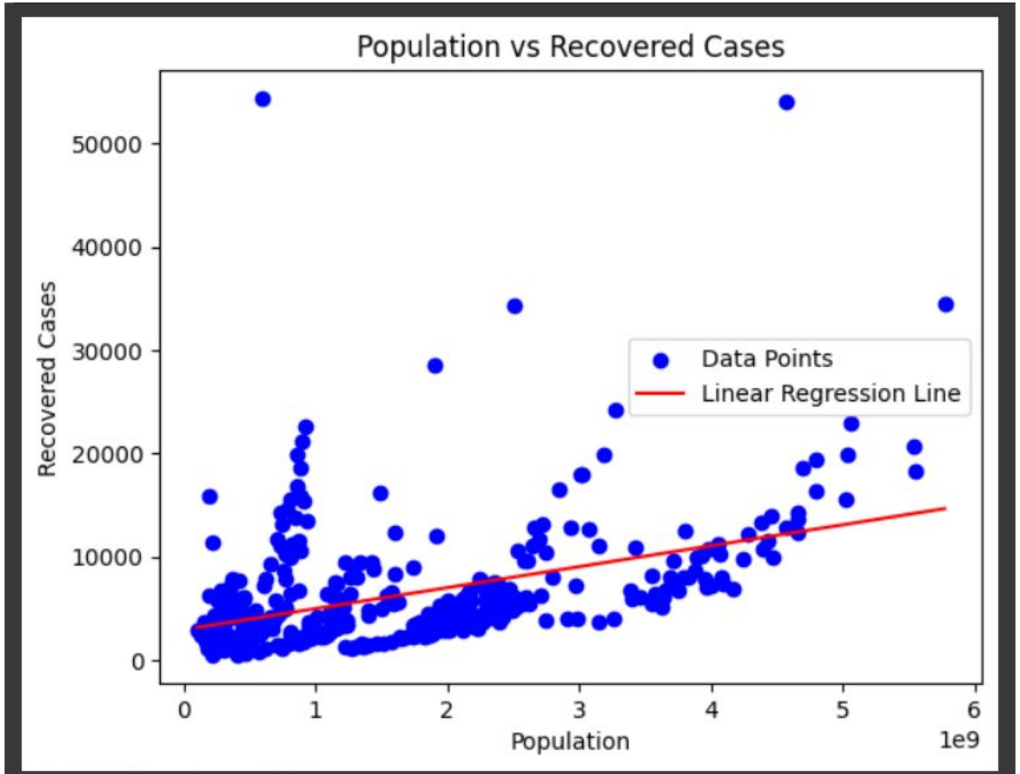
```
x = np.array(df.groupby('county')['population'].sum()).reshape(-1,1) #total population by county
y = np.array(df.groupby('county')['recovered'].sum()).reshape(-1,1) #total number of recovered cases by county
```

2. Prediction

Then we use the two values to find a line that fits the data

3. Analysis

Finally we evaluate the model using R^2 and a visual graph of the data points and line



```
R-squared score: 0.18363
```