CM3400 Report

Introduction

In 2019 a new form of Coronovirus was discovered, called SARS-CoV-2. It causes mild symptons in most people but in the elderly and those with underlying health conditions it can cause much more serious illnesses. It is highly contagious and can be transmitted from person to person. In 2020 the outbreak developed into a world-wide pandemic, more than 3 million people as of April 2021 have died after contracting the virus. I will be taking a deeper look at some of the statistics of the corona virus outbreak, as well as seeing if it is possible to model the death rate and mortality.

Dataset

The data set consists of profiles of mamy countries. It collates information on cases, deaths and government responses for each country in relation to the Covid-19 outbreak. The data covers a year, broken down into 3 month periods, from February 2020 to Februrary 2021.

Load packages and data set

Firstly loading packages and data set into python and checking size of data set.

In [54]:
```
import pandas as pd
import seaborn
import matplotlib.pyplot as plt
import statistics as stat
import numpy as np
from datetime import datetime
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import sklearn.metrics as metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn import preprocessing
from sklearn.ensemble import RandomForestRegressor
```

In [55]:
```
df = pd.read_csv("C:/Users/rossm/CM3400/covid.csv")
print(df.shape)
df.head(5)
```

```
(525, 23)
```

Out[55]:

	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_older
0	Afghanistan	Asia	May-20	390.641	6.602	84.26	54.422	18.6	2.581	1.337
1	Albania	Europe	May-20	394.398	11.120	86.11	104.871	38.0	13.188	8.643
2	Algeria	Africa	May-20	214.202	14.868	76.85	17.348	29.1	6.211	3.857
3	Argentina	South America	May-20	372.823	11.904	90.74	16.177	31.9	11.198	7.441
4	Australia	Oceania	May-20	281.374	4.000	63.43	3.202	37.9	15.504	10.129

5 rows × 23 columns

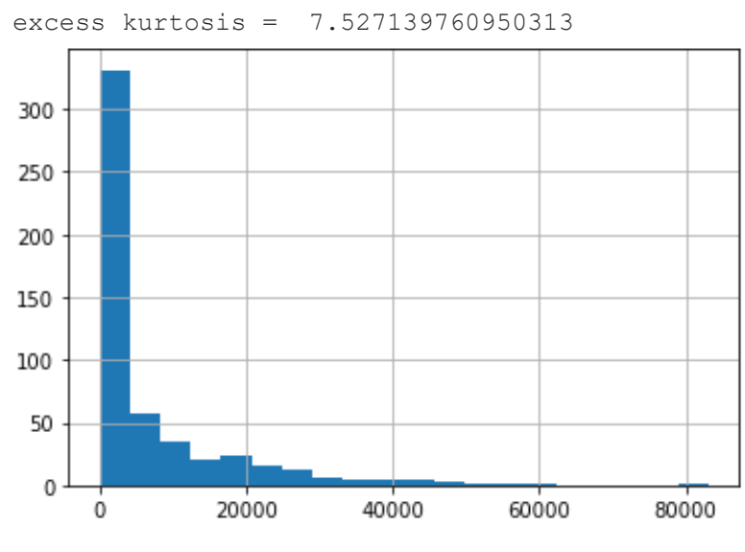
23 variables on 525 different country/date combinations.

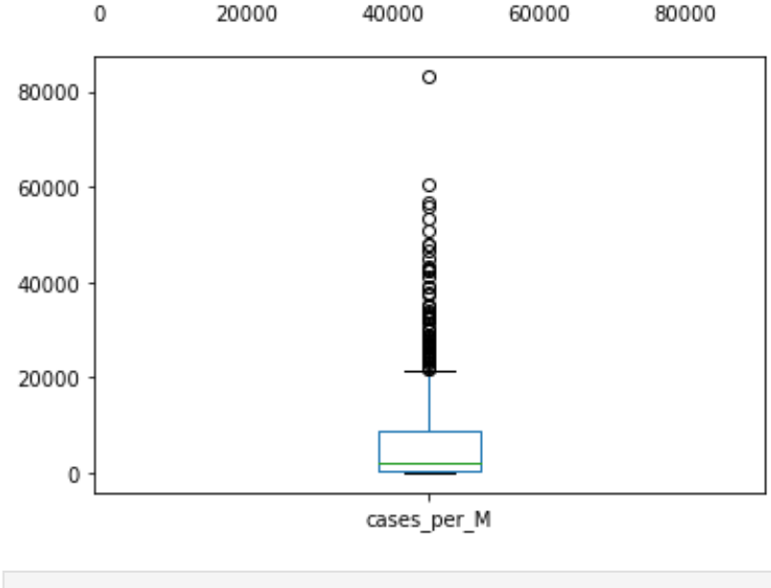
Statistics of individual variables

Now to look at the number of confirmed cases of coronavius per million people in each country.

In [56]:
```
x = df['cases_per_M']
print("Using pandas module functions")
print("  n = ", x.count())
print("Measures of central location")
print("mean = ", x.mean())
print("median = ", x.median())

print("Measures of spread")
print("range = ", x.max()-x.min())
q = x.quantile([0.25,0.5,0.75])
print("quartiles = ", q)
print("IQR = ", x.quantile(0.75) - x.quantile(0.25))
print("stdev = ", x.std())
print("pop stdev = ", x.std(ddof = 0))

print("Measures of shape")
print("skewness = ", x.skew())
print("excess kurtosis = ", x.kurt())

x.hist(bins = 20)
plt.show()
x.plot.box()
plt.show()
```

```
Using pandas module functions
  n =  525
Measures of central location
mean =  7117.880066666669
median =  1943.832
Measures of spread
range =  83074.35399999999
quartiles =  0.25     322.346
0.50    1943.832
0.75    8856.786
Name: cases_per_M, dtype: float64
IQR =  8534.44
stdev =  11393.095143653863
pop stdev =  11382.239404987053
Measures of shape
skewness =  2.4961269965529684
excess kurtosis =  7.527139760950313
```





In [57]:
```
dfcases = df[df['cases_per_M'] > 40000]
dfcases.sort_values(by = ['cases_per_M'], ascending=False)
```

Out[57]:

	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_olde
421	Czechia	Europe	Feb-21	83076.430	1434.497	69.44	137.176	43.3	19.027	11.58
499	Slovenia	Europe	Feb-21	60455.561	1205.427	69.44	102.619	44.5	19.062	12.93
429	Estonia	Europe	Feb-21	56719.026	468.136	41.67	31.033	42.7	19.452	13.49
452	Israel	Asia	Feb-21	56092.623	366.932	57.41	402.606	30.6	11.733	7.35
465	Lithuania	Europe	Feb-21	53256.246	1027.076	66.67	45.135	43.5	19.002	13.77
491	Portugal	Europe	Feb-21	50686.560	1196.661	87.96	112.371	46.2	21.502	14.92
518	United States	North America	Feb-21	47971.498	803.773	68.06	35.608	38.3	15.413	9.73
506	Sweden	Europe	Feb-21	47742.065	642.720	69.44	24.718	41.0	19.985	13.43
288	Czechia	Europe	Nov-20	46566.518	734.990	69.44	137.176	43.3	19.027	11.58
332	Luxembourg	Europe	Nov-20	44814.817	314.708	60.19	231.447	39.7	14.312	9.84
462	Lebanon	Asia	Feb-21	43292.288	652.851	92.59	594.561	31.1	8.514	5.43
498	Slovakia	Europe	Feb-21	42788.329	1434.160	71.30	113.128	41.2	15.070	9.16
270	Belgium	Europe	Nov-20	42461.199	582.417	63.89	375.564	41.8	18.571	12.84
486	Panama	North America	Feb-21	42360.099	679.063	74.07	55.133	29.7	7.918	5.03
461	Latvia	Europe	Feb-21	41102.172	832.891	56.48	31.212	43.9	19.754	14.13

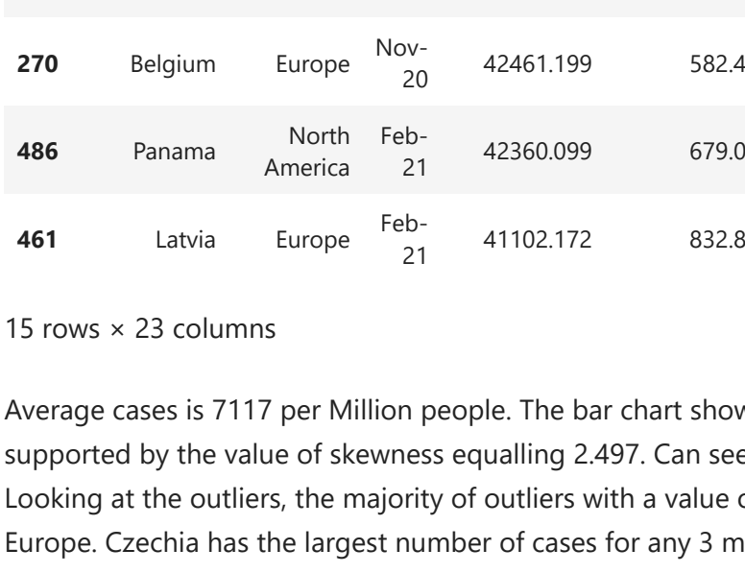
15 rows × 23 columns

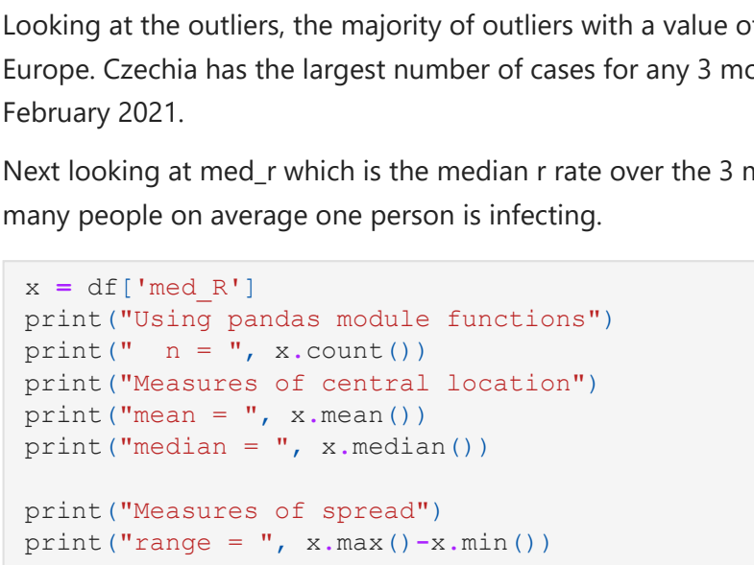
Average cases is 7117 per Million people. The bar chart shows that cases per million of people is stronged positvely skewed, this is supported by the value of skewness equalling 2.497. Can see that the majority of cases is between 0 and 4000 with numerous large outliers. Looking at the outliers, the majority of outliers with a value of cases per million people greater than 40,000 come from coutries inside Europe. Czechia has the largest number of cases for any 3 month peroid with around 83,000 cases per million from December 2020 to February 2021.

Next looking at med_r which is the median r rate over the 3 month period in each country. The r rate (reproduction number) describes how many people on average one person is infecting.

In [58]:
```
x = df['med_R']
print("Using pandas module functions")
print("  n = ", x.count())
print("Measures of central location")
print("mean = ", x.mean())
print("median = ", x.median())

print("Measures of spread")
print("range = ", x.max()-x.min())
q = x.quantile([0.25,0.5,0.75])
print("quartiles = ", q)
print("IQR = ", x.quantile(0.75) - x.quantile(0.25))
print("stdev = ", x.std())
print("pop stdev = ", x.std(ddof = 0))

print("Measures of shape")
print("skewness = ", x.skew())
print("excess kurtosis = ", x.kurt())

x.hist(bins = 20)
plt.show()
x.plot.box()
plt.show()
```

```
Using pandas module functions
  n =  525
Measures of central location
mean =  1.0392190476190468
median =  1.05
Measures of spread
range =  1.8
quartiles =  0.25    0.95
0.50    1.05
0.75    1.15
Name: med_R, dtype: float64
IQR =  0.19999999999999996
stdev =  0.22085270128986442
pop stdev =  0.2206422651284147
Measures of shape
skewness =  -1.316562912322257
excess kurtosis =  5.021591227384064
```





In [59]:
```
df.sort_values(by = ['med_R'], ascending=False).head(5)
```

Out[59]:

	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_older
84	Nepal	Asia	May-20	53.918	0.241	92.59	204.430	25.0	5.809	3.212
74	Malawi	Africa	May-20	14.689	0.157	60.19	197.519	18.1	2.979	1.783
112	Tajikistan	Asia	May-20	410.479	4.718	44.44	64.281	23.3	3.466	2.155
94	Peru	South America	May-20	4988.347	136.571	89.81	25.129	29.1	7.151	4.455
38	Ethiopia	Africa	May-20	10.186	0.079	80.56	104.957	19.8	3.526	2.063

5 rows × 23 columns

In [60]:
```
df.sort_values(by = ['med_R'], ascending=True).head(5)
```

Out[60]:

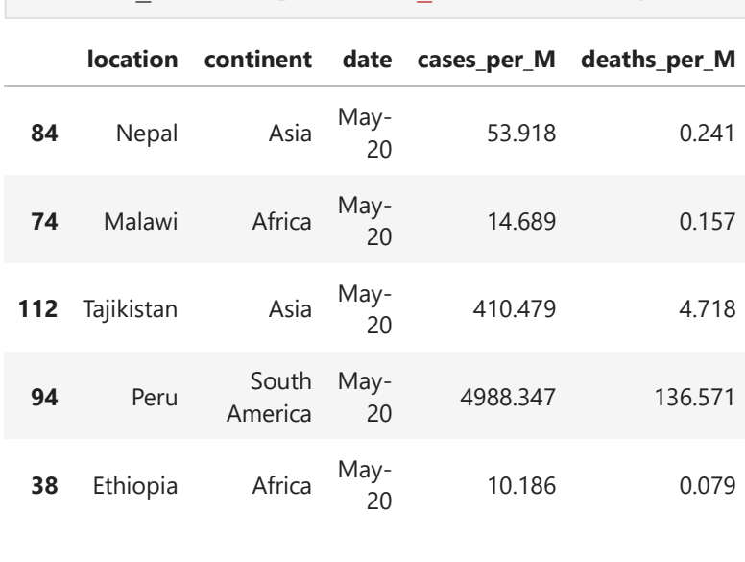
	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_older
508	Tajikistan	Asia	Feb-21	116.800	0.419	16.67	64.281	23.3	3.466	2.155
147	Brunei	Asia	Aug-20	6.858	2.285	40.74	81.347	32.4	4.591	2.382
116	Trinidad and Tobago	North America	May-20	82.173	5.001	87.04	266.886	36.2	10.014	5.819
16	Brunei	Asia	May-20	320.012	2.286	52.78	81.347	32.4	4.591	2.382
7	Bahamas	North America	May-20	256.835	25.429	92.59	39.497	34.3	8.996	5.200

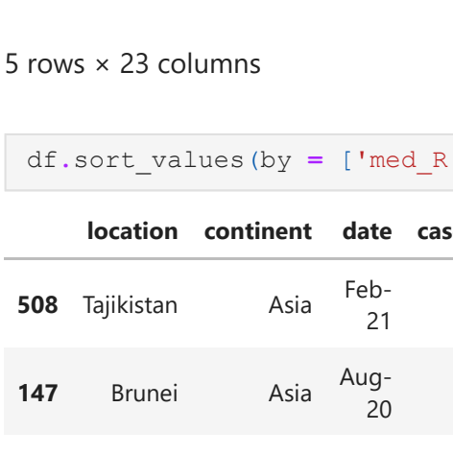
5 rows × 23 columns

The average median r rate is 1.03, meaning, on average across the year, 1 person was infecting just above 1 other person. From looking at bar chart and the value of skewness, can see that values are slightly negatively skewed. Most values between 1 and 1.25. Some outliers over 1.45 and below 0.75. Outliers below 0.12 could negligible as there were cases in those countries during that period, so the low r rate could be down to the limited contact_tracing. Outliers at the top include Nepal which had the largest median R value for any 3 month perioid of 1.8, meaning that every infected person would on average infect another 1.8 people.

In [61]:
```
print("  n = ", df['mortality'].count())

counts = df['mortality'].value_counts()
print(counts)

df['mortality'].value_counts().plot.bar()
plt.show()
df['mortality'].value_counts().plot.pie(autopct = '%1.1f')
plt.show()
```

```
  n =  525
Low     304
High    221
Name: mortality, dtype: int64
```

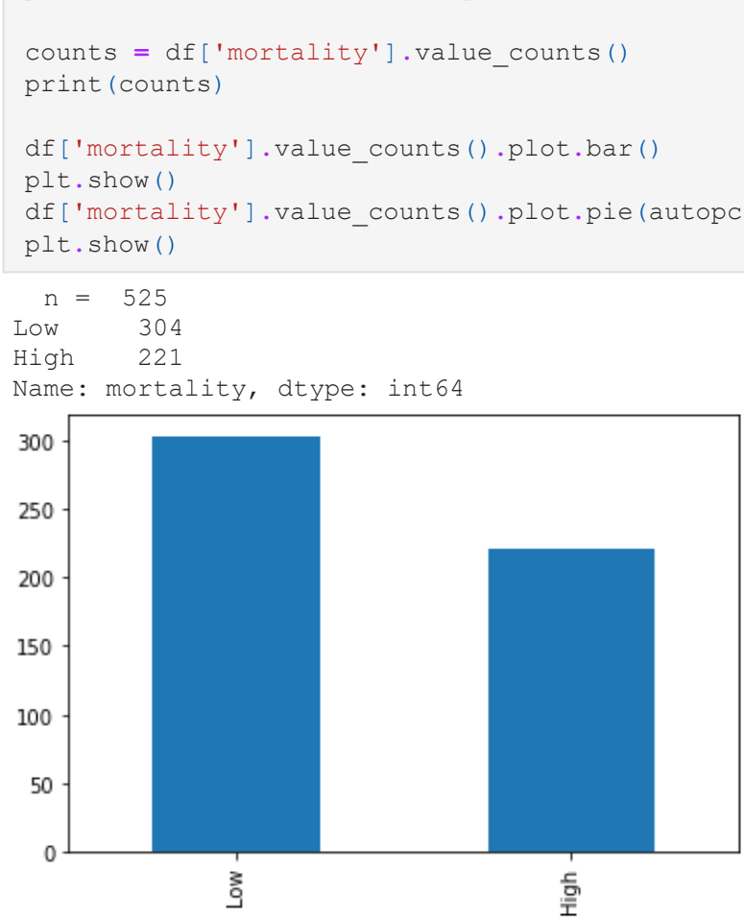


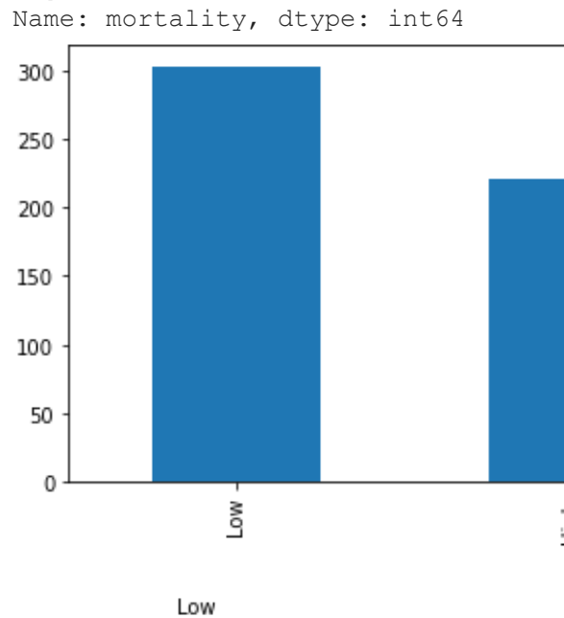


Over the course of the year most countries had a low mortality rate during the 3 month perioid. Low mortality rate of 57.9% comprated to a high mortaltiy rate of 42.1%.

In [62]:
```
print("  n = ", df['international_travel_controls'].count())

counts = df['international_travel_controls'].value_counts()
print(counts)

df['international_travel_controls'].value_counts().plot.bar()
plt.show()
df['international_travel_controls'].value_counts().plot.pie(autopct = '%1.1f')
plt.show()
```

```
  n =  525
Ban               264
Quarantine        152
BorderClosure      54
Screening          51
None                4
Name: international_travel_controls, dtype: int64
```

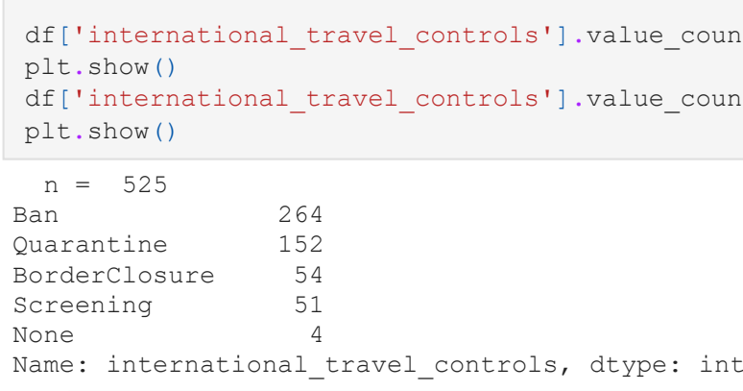


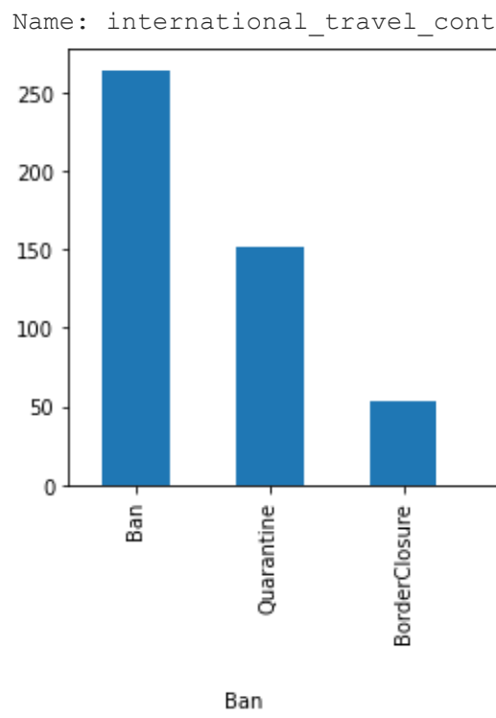


Majority of countries imposed bans on countries, that were considered to be high risk, from international travel. Whereas 10.3% imposed complete border closure with no travel in or out. Less than 1% imposed no travel restrictions during the 3 month peroid.

In [63]:
```
print("  n = ", df['workplace_closures'].count())

counts = df['workplace_closures'].value_counts()
print(counts)

df['workplace_closures'].value_counts().plot.bar()
plt.show()
df['workplace_closures'].value_counts().plot.pie(autopct = '%1.1f')
plt.show()
```

```
  n =  525
Strict    315
Some      147
None       63
Name: workplace_closures, dtype: int64
```





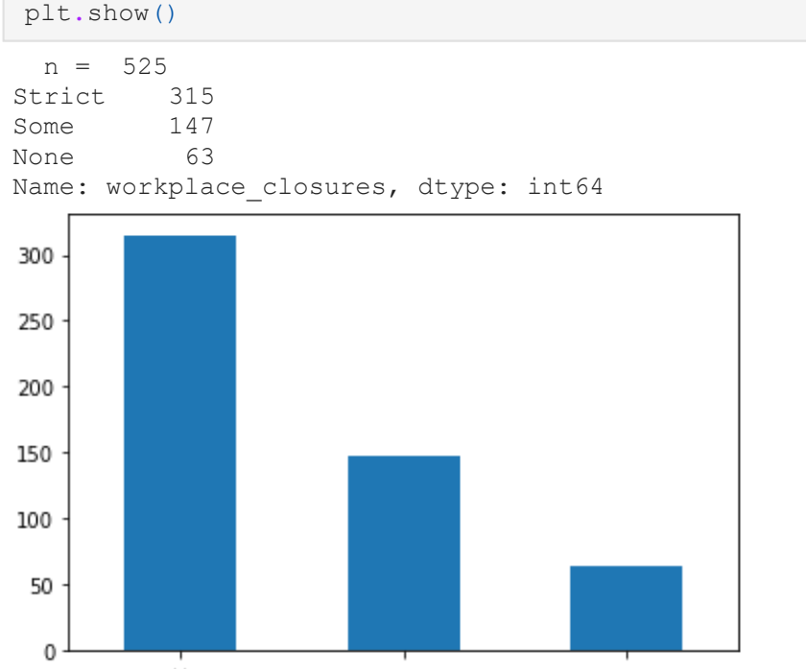
Two thirds of countries during a 3 month period imposed strict workplace closures. Whilst 12% imposed none, with 28% imposing some restricts.

Filtering data

In [64]:
```
t6 = df[['cases_per_M','continent','date']]
t6.boxplot('cases_per_M', by = 'continent')
ir = t6.sort_values('cases_per_M').groupby('continent')
ir.cases_per_M.describe()
```

Out[64]:

	count	mean	std	min	25%	50%	75%	max
continent								
Africa	125.0	1765.336080	4000.209981	8.095	115.71800	322.346	1165.90200	32245.271
Asia	140.0	5329.409607	9251.079090	2.076	237.38000	1315.724	6336.67325	56092.623
Europe	147.0	13718.374891	15752.497741	174.955	1492.50350	6246.246	22788.93400	83076.430
North America	60.0	6605.486000	9205.816195	82.173	619.64500	3543.546	7969.69900	47971.498
Oceania	7.0	227.907143	246.628763	48.706	70.71400	82.078	296.52800	730.082
South America	46.0	7729.936109	6845.526468	53.032	1682.02575	6294.498	13220.78675	24782.046



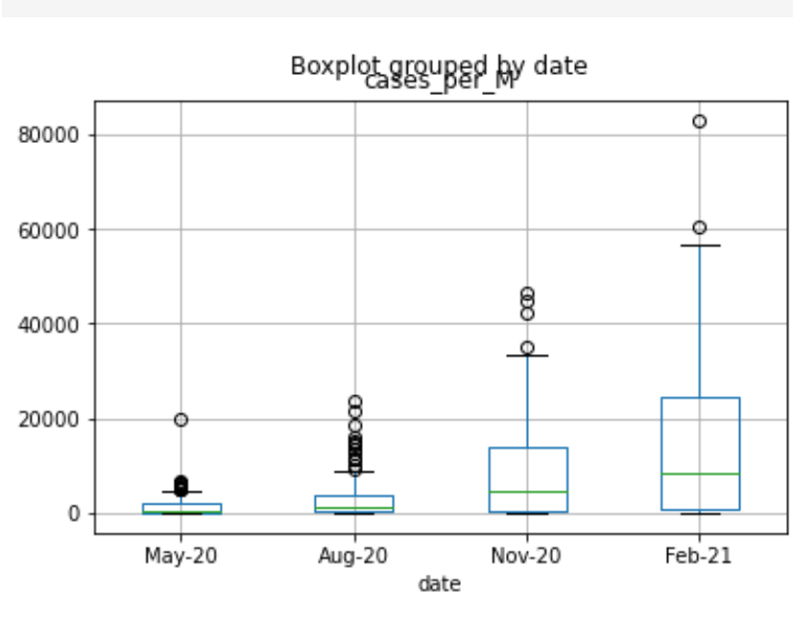
Europe has the highest average cases per million people (13,700), as well has the highest value and the largest spread of values. Oceania has the lowest spread and average (227), could be down to there only being data for seven 3-month periods. Asia and Africa have a number of outliers, whereas South America has none.

In [65]:
```
t6['date'] = pd.Categorical(df['date'], categories=['May-20','Aug-20','Nov-20','Feb-21'])
boxplot = t6.boxplot('cases_per_M', by = 'date')
ir2 = t6.sort_values('cases_per_M').groupby('date')
ir2.cases_per_M.describe()
```

```
<ipython-input-65-a8d11abc3dde>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#ret
urning-a-view-versus-a-copy
  t6['date'] = pd.Categorical(df['date'], categories=['May-20','Aug-20','Nov-20','Feb-21'])
```

Out[65]:

	count	mean	std	min	25%	50%	75%	max
date								
May-20	129.0	1408.241124	2371.120890	2.928	138.74200	387.9010	1972.37800	19752.105
Aug-20	130.0	3214.191569	4554.746620	4.008	461.65925	1279.7225	3825.04475	23844.855
Nov-20	131.0	9128.803321	10578.782893	2.076	427.99350	4769.3960	14077.58450	46566.518
Feb-21	135.0	14381.524378	16443.820350	5.934	952.28600	8541.4000	24568.58800	83076.430



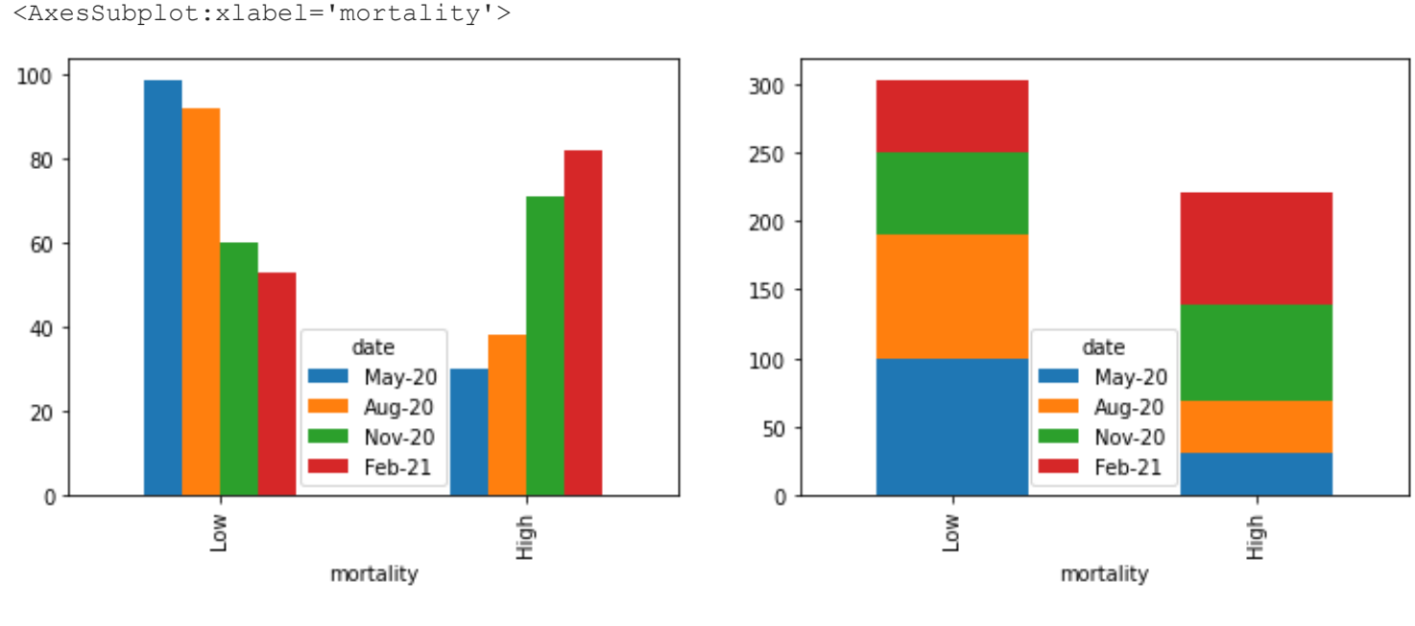
The average number of cases per million people increases in with time. Averages cases from February 2020 to May 2020 was 1408 per million people. Increased to 14381 per million people for December 2020 to February 2021. Three months preceding February 2021 has the largest spread of values. Lots of outliers in August 2020 where supposedly some countries were hit by an outbreak early than others, by February 2021 the virus has spread more evenly across the world.

Cross-Tabulation

In [66]:
```
df['date'] = pd.Categorical(df['date'], categories=['May-20','Aug-20','Nov-20','Feb-21'])
df['mortality'] = pd.Categorical(df['mortality'], categories=['Low','High'])
ct1 = pd.crosstab(df['mortality'], df['date'])

print(ct1)

fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ct1.plot.bar(ax = ax[0])
ct1.plot.bar(stacked = True, ax = ax[1])
```

```
date       May-20  Aug-20  Nov-20  Feb-21
mortality
Low            99      92      60      53
High           30      38      71      82
```

Out[66]:
```
<AxesSubplot:xlabel='mortality'>
```

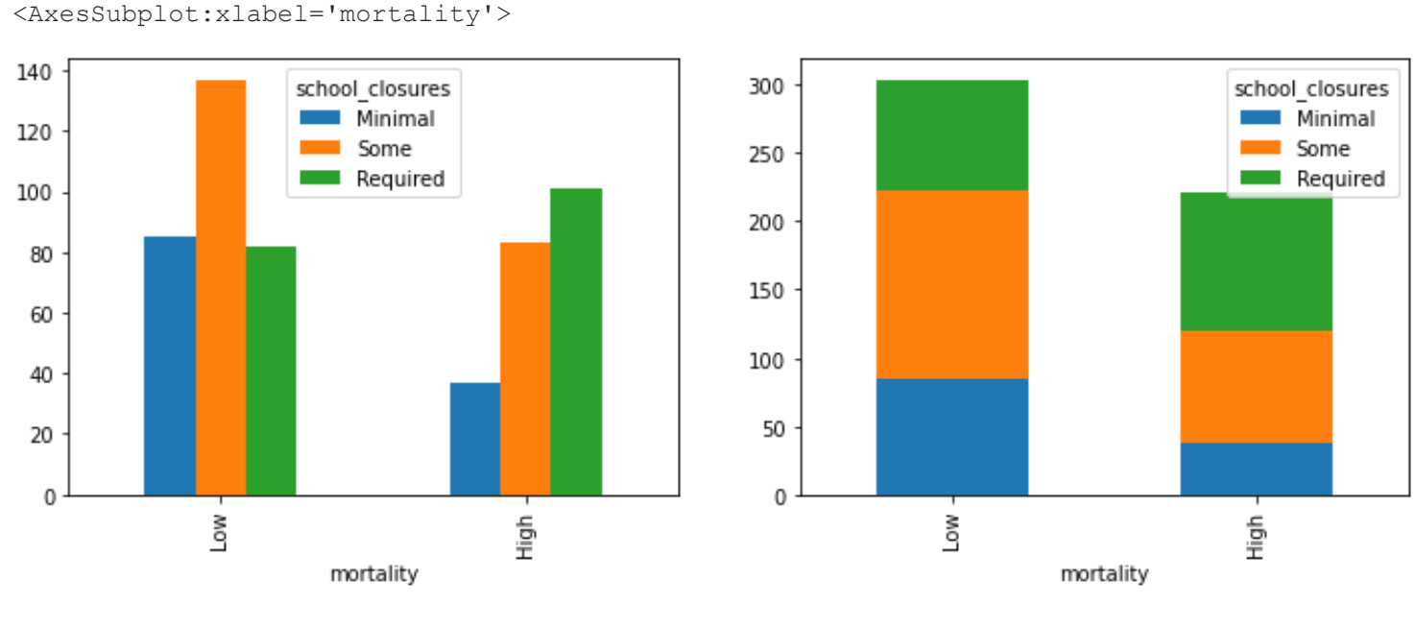


99 countries had a low mortality rate in the months preceding May 2020, whereas only 30 had a high mortality rate. Contries with low mortality rate decreases with each 3 month peroid, to 92 in August 2020, then 60 in November 2020 and finally 53 in February 2021. Countries with high mortality increases, to 82 in February 2021. Shows trend in increase in high mortality rate with time and decrease in low mortality rate with time.

In [67]:
```
df['school_closures'] = pd.Categorical(df['school_closures'], categories=['Minimal','Some','Required'])
ct2 = pd.crosstab(df['mortality'], df['school_closures'])
print(ct2)

fig, ax = plt.subplots(1, 2, figsize=(12, 4))
ct2.plot.bar(ax = ax[0])
ct2.plot.bar(stacked = True, ax = ax[1])
```

```
school_closures  Minimal  Some  Required
mortality
Low                   85   137        82
High                  37    83       101
```

Out[67]:
```
<AxesSubplot:xlabel='mortality'>
```



When mortality rate is high largest percentage of goverments impose school closures, 101 out of 221, with only 37 out of 221 imposing minimal closures.

Data Processing

Creating a new datetime variable.

In [68]:
```
df['datetime'] = pd.to_datetime(df['date'], format = '%b-%y')
df.dtypes
```

Out[68]:
```
location                          object
continent                         object
date                            category
cases_per_M                      float64
deaths_per_M                     float64
stringency_index                 float64
population_density               float64
median_age                       float64
aged_65_older                    float64
aged_70_older                    float64
gdp_per_capita                   float64
life_expectancy                  float64
human_development_index          float64
med_R                            float64
max_R                            float64
workplace_closures                object
contact_tracing                   object
testing_policy                    object
cancel_public_events              object
facial_coverings                  object
international_travel_controls     object
school_closures                 category
mortality                       category
datetime                        category
dtype: object
```

Creating a new varaible that splits median age of the country into 3 different brackets.

In [69]:
```
ageband = pd.cut(df.median_age, bins = [0,20,35,50],
                 labels =['young','mid','older'])
df = df.assign(AgeCategory = ageband)
df.head(5)
```

Out[69]:

	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_older
0	Afghanistan	Asia	May-20	390.641	6.602	84.26	54.422	18.6	2.581	1.337
1	Albania	Europe	May-20	394.398	11.120	86.11	104.871	38.0	13.188	8.643
2	Algeria	Africa	May-20	214.202	14.868	76.85	17.348	29.1	6.211	3.857
3	Argentina	South America	May-20	372.823	11.904	90.74	16.177	31.9	11.198	7.441
4	Australia	Oceania	May-20	281.374	4.000	63.43	3.202	37.9	15.504	10.129

5 rows × 25 columns

Removing all missing values from data set before modelling.

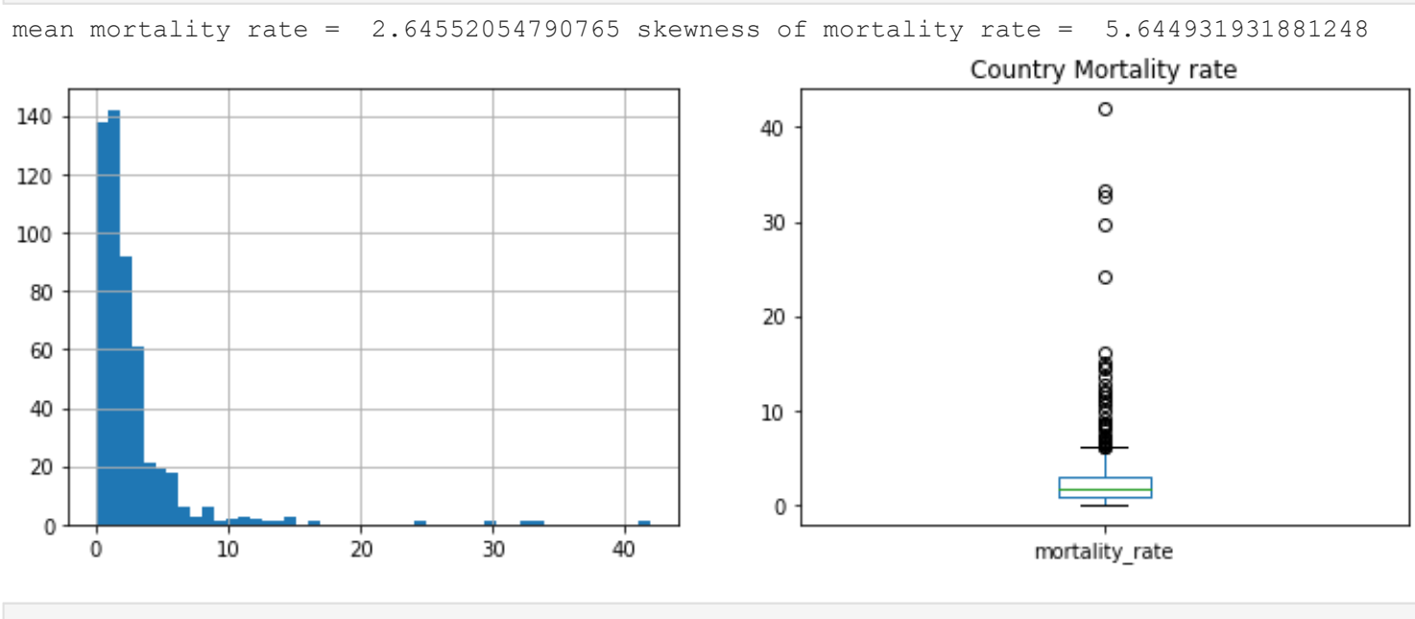
In [70]:
```
print("Show count of missing values\n\r", df.notna().all())

df.dropna(inplace = True)
print("Show count of missing values\n\r", df.notna().all())
```

```
Show count of missing values
 location                         True
continent                        True
date                             True
cases_per_M                      True
deaths_per_M                     True
stringency_index                 True
population_density               True
median_age                       True
aged_65_older                    True
aged_70_older                    True
gdp_per_capita                   True
life_expectancy                  True
human_development_index          True
med_R                            True
max_R                            True
workplace_closures               True
contact_tracing                  True
testing_policy                   True
cancel_public_events             True
facial_coverings                 True
international_travel_controls    True
school_closures                  True
mortality                        True
datetime                         True
AgeCategory                      True
dtype: bool
Show count of missing values
 location                         True
continent                        True
date                             True
cases_per_M                      True
deaths_per_M                     True
stringency_index                 True
population_density               True
median_age                       True
aged_65_older                    True
aged_70_older                    True
gdp_per_capita                   True
life_expectancy                  True
human_development_index          True
med_R                            True
max_R                            True
workplace_closures               True
contact_tracing                  True
testing_policy                   True
cancel_public_events             True
facial_coverings                 True
international_travel_controls    True
school_closures                  True
mortality                        True
datetime                         True
AgeCategory                      True
dtype: bool
```

Creating a new variable for mortality rate. Deaths per million divided by cases per million.

In [71]:
```
mr = 100 * df['deaths_per_M']/df['cases_per_M']

df = df.assign(mortality_rate = mr)
df['mortality_rate'].describe()

fig, ax = plt.subplots(1, 2, figsize=(12, 4))
df['mortality_rate'].hist(bins = 47, ax = ax[0])
df['mortality_rate'].plot.box(title = 'Country Mortality rate', ax = ax[1])
print("mean mortality rate = ", df['mortality_rate'].mean(), "skewness of mortality rate = ", df['mortality_rat
```

```
mean mortality rate =  2.64552054790765 skewness of mortality rate =  5.644931931881248
```



In [72]:
```
df.sort_values(by = ['mortality_rate'], ascending=False).head(5)
```

Out[72]:

	location	continent	date	cases_per_M	deaths_per_M	stringency_index	population_density	median_age	aged_65_older	aged_70_older
24	China	Asia	May-20	2.928	1.227	81.94	147.674	38.7	10.641	5.929
147	Brunei	Asia	Aug-20	6.858	2.285	40.74	81.347	32.4	4.591	2.382
387	Yemen	Asia	Nov-20	3.990	1.307	16.67	53.508	20.3	2.922	1.583
256	Yemen	Asia	Aug-20	54.818	16.295	31.48	53.508	20.3	2.922	1.583
126	Yemen	Asia	May-20	10.795	2.615	58.33	53.508	20.3	2.922	1.583

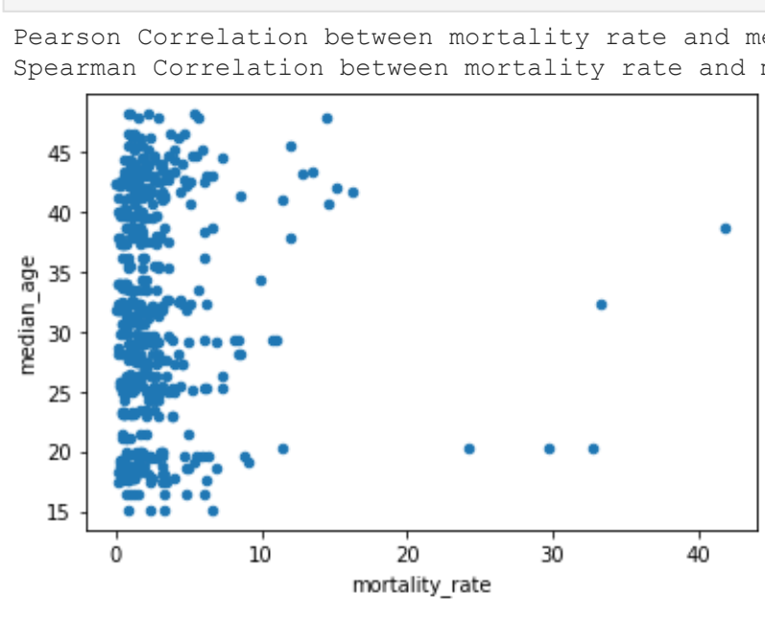
5 rows × 26 columns

Average mortality rate of 2.65%. Very strongly positively skewed. Majority of values between 0% and 4%. Some large outliers potentially skewing mean. China had mortaltiy rate of 41.9% in 3 months preceding May 2020.

Correlation of mortality rate against median age.

In [73]:
```
df.plot.scatter(x= 'mortality_rate', y ='median_age')
print("Pearson Correlation between mortality rate and median age is ", df.mortality_rate.corr(df['median_age'])
print("Spearman Correlation between mortality rate and median age is ", df.mortality_rate.corr(df['median_age']
```

```
Pearson Correlation between mortality rate and median age is  0.012556385376079986
Spearman Correlation between mortality rate and median age is  0.036369899578773686
```



From graph and Person correlation coefficent can determine that there is likely no relationship between the median age of a countries population and the mortality rate.

Data Modelling

Linear Models

Creating a linear regression model of deaths per million people against all other variables except location. Splitting the data set into two equal train/test parts.

In [74]:
```
y = df['deaths_per_M']

X = df.drop(columns= ['deaths_per_M','location','mortality','mortality_rate','datetime','median_age'])
X = pd.get_dummies(X)

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.5, random_state=42)

lr = LinearRegression()
model = lr.fit(X_train,y_train)

y_pred = model.predict(X_test)
print("R^2(test) = ", metrics.r2_score(y_test, y_pred))
print("MSE(test) = ", metrics.mean_squared_error(y_test, y_pred))
print("MAE(test) = ", metrics.mean_absolute_error(y_test, y_pred))
```

```
R^2(test) =  0.5767327161126549
MSE(test) =  19986.905093522524
MAE(test) =  95.19877932706042
```

The model can explain 57.7% of variability in the data. Suggesting that there is more factors affecting deaths per million people that is not captured in the data.

Creating a random forrest regression model of deaths per million people against all other varaibles. Again splitting data 50/50 into training and test sets.

```
> X_DT_Tomorrow_pca_df=
> X_DT_Tomorrow_pca_df.shape
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=1)
> X_DT_Tomorrow_pca=Tomorrow_pca_df[X_DT_pca_columns]
> X_DT_Tomorrow_pca=X_DT_Tomorrow_pca[X_DT_pca_columns]
> y_pred = DTtreeclf.predict(X_test)
> y_pred
array([0, 0, 0, ..., 0, 0, 0])
> print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.9999333333333333
> print("Precision:",metrics.precision_score(y_test, y_pred))
Precision: 0.9999333333333333
> print("Recall:",metrics.recall_score(y_test, y_pred))
Recall: 1.0
> y_pred_DT=DTtreeclf.predict(X_DT_Tomorrow_pca)
> y_pred_DT
array([0, 0, 0, ..., 0, 0, 0])
```

Model explains 99.9% of the variability in the data

There the outlook forest regressor can account for more of the variability in the data than the linear regression model

Classification models:

Creating decision tree classifier, order of deaths per million people sorted of other condition, Again splitting data 70/30 into training and testing data

```
> X_DT_Tomorrow_pca=
> X_DT_Tomorrow_pca.shape
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=1)
> X_DT_Tomorrow_pca=Tomorrow_pca_df[X_DT_pca_columns]
> X_DT_Tomorrow_pca=X_DT_Tomorrow_pca[X_DT_pca_columns]
> y_pred = DTtreeclf.predict(X_test)
> y_pred
array([0, 0, 0, ..., 0, 0, 0])
> print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 1.0
> print("Precision:",metrics.precision_score(y_test, y_pred))
Precision: 1.0
> print("Recall:",metrics.recall_score(y_test, y_pred))
Recall: 1.0
> y_pred_DT=DTtreeclf.predict(X_DT_Tomorrow_pca)
> y_pred_DT
array([0, 0, 0, ..., 0, 0, 0])
```

Model accuracy score 100% of mortality classifications. 100 were wrongly predicted as low. 0 wrongly predicted as high

```
> X_DT_Tomorrow_pca=
> X_DT_Tomorrow_pca.shape
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_DT_pca_columns=X_DT_Tomorrow_pca_df.columns
> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state=1)
> X_DT_Tomorrow_pca=Tomorrow_pca_df[X_DT_pca_columns]
> X_DT_Tomorrow_pca=X_DT_Tomorrow_pca[X_DT_pca_columns]
> y_pred = DTtreeclf.predict(X_DT_Tomorrow_pca)
> y_pred
array([0, 0, 0, ..., 0, 0, 0])
> print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.9999333333333333
> print("Precision:",metrics.precision_score(y_test, y_pred))
Precision: 0.9999333333333333
> print("Recall:",metrics.recall_score(y_test, y_pred))
Recall: 1.0
> y_pred_DT=DTtreeclf.predict(X_DT_Tomorrow_pca)
> y_pred_DT
array([0, 0, 0, ..., 0, 0, 0])
```

Model accuracy score 99.9% of mortality classifications. 100 were wrongly predicted as low. 0 wrongly predicted as high

Comparing the two models, the decision tree model is more accurate than the K-nearest neighbours model