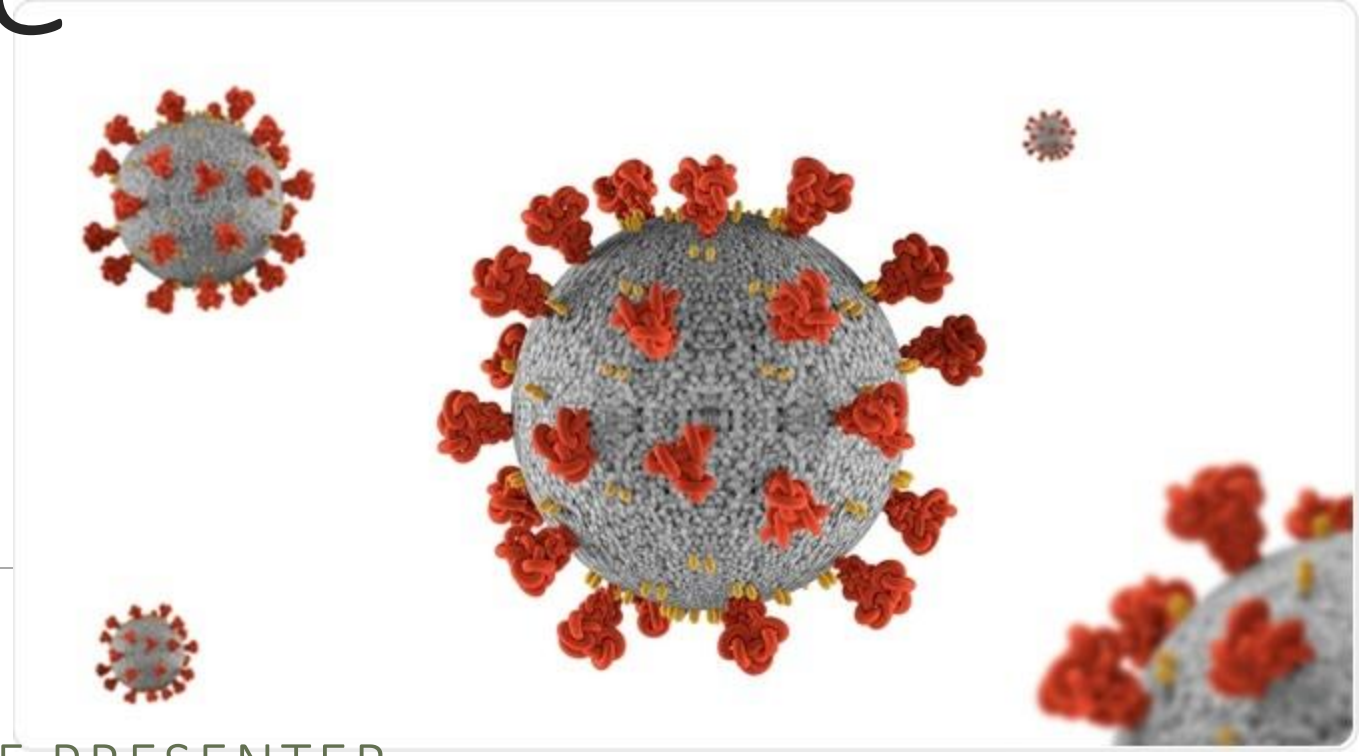
Title



NAME OF PRESENTER

Dataset Description

This dataset is a collection of the COVID-19 data maintained by *Our World in Data*. They update it daily and will keep updating throughout the duration of the COVID-19 pandemic. It includes the data of 65 different variables. The Dataset contains information from 22 February 2020 to till now. It consists of 132644 records with 65 different columns such as total_cases, new_cases, total_deaths etc.

Project Objective

Analyze the COVID-19 dataset to explore meaningful information and train a machine learning model so that to predict new cases and new death at early stage. Government will be beneficial from this project to take suitable action at early stage after prediction.

Dataset Description

Overview

Overview | Alerts 293 | Reproduction

Dataset statistics

Number of variables	65
Number of observations	132644
Missing cells	3848452
Missing cells (%)	44.6%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	65.8 MiB
Average record size in memory	520.0 B

Variable types

Categorical	5
Numeric	60

Dataset Description

Total Columns

0	iso_code
1	continent
2	location
3	date
4	total_cases
5	new_cases
6	new_cases_smoothed
7	total_deaths
8	new_deaths
9	new_deaths_smoothed
10	total_cases_per_million
11	new_cases_per_million
12	new_cases_smoothed_per_million
13	total_deaths_per_million
14	new_deaths_per_million
15	new_deaths_smoothed_per_million
16	reproduction_rate
17	icu_patients
18	icu_patients_per_million
19	hosp_patients
20	hosp_patients_per_million
21	weekly_icu_admissions
22	weekly_icu_admissions_per_million
23	weekly_hosp_admissions
24	weekly_hosp_admissions_per_million
25	new_tests
26	total_tests
27	total_tests_per_thousand
28	new_tests_per_thousand
29	new_tests_smoothed
30	new_tests_smoothed_per_thousand
31	positive_rate
32	tests_per_case
33	tests_units
34	total_vaccinations
35	people_vaccinated
36	people_fully_vaccinated
37	total_boosters
38	new_vaccinations
39	new_vaccinations_smoothed
40	total_vaccinations_per_hundred
41	people_vaccinated_per_hundred
42	people_fully_vaccinated_per_hundred
43	total_boosters_per_hundred
44	new_vaccinations_smoothed_per_million
45	stringency_index
46	population
47	population_density
48	median_age
49	aged_65_older
50	aged_70_older
51	gdp_per_capita
52	extreme_poverty
53	cardiovasc_death_rate
54	diabetes_prevalence
55	female_smokers
56	male_smokers
57	handwashing_facilities
58	hospital_beds_per_thousand
59	life_expectancy
60	human_development_index
61	excess_mortality_cumulative_absolute
62	excess_mortality_cumulative
63	excess_mortality
64	excess_mortality_cumulative_per_million

Dataset Description

total_cases

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
MISSING

Distinct	73232
Distinct (%)	58.4%
Missing	7234
Missing (%)	5.5%
Infinite	0
Infinite (%)	0.0%
Mean	2038985.127

Minimum	1
Maximum	251941758
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	1.0 MiB

Toggle details

Dataset Description

total_deaths

Real number ($\mathbb{R}_{\geq 0}$)

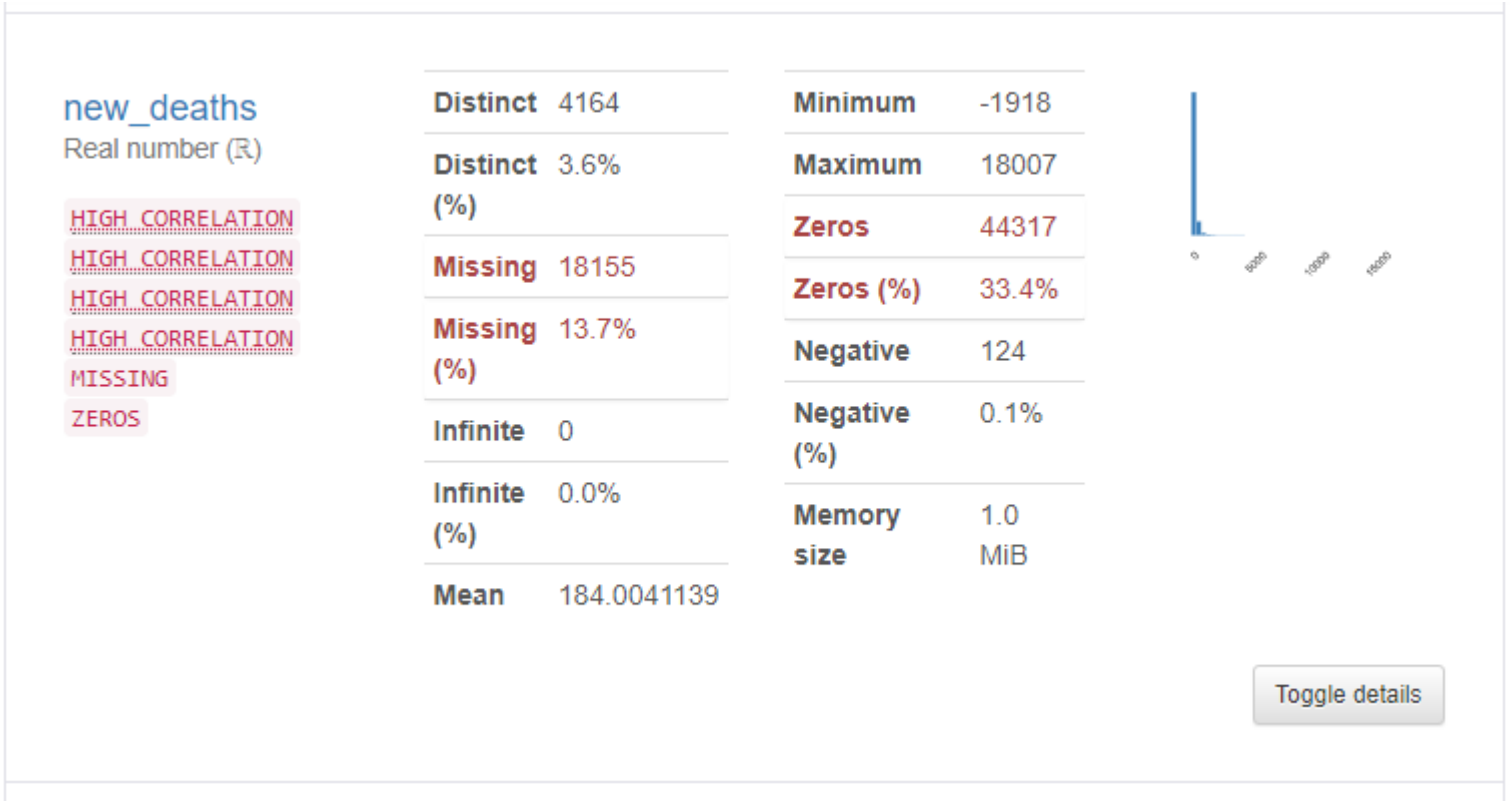
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
MISSING

Distinct	29084
Distinct (%)	25.4%
Missing	18351
Missing (%)	13.8%
Infinite	0
Infinite (%)	0.0%
Mean	50452.49547

Minimum	1
Maximum	5079990
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	1.0 MiB

Toggle details

Dataset Description



Dataset Description

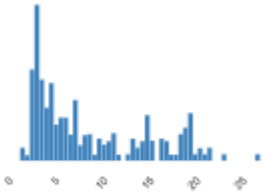
aged_65_older

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
MISSING

Distinct	188
Distinct (%)	0.2%
Missing	19485
Missing (%)	14.7%
Infinite	0
Infinite (%)	0.0%
Mean	8.744577426

Minimum	1.144
Maximum	27.049
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	1.0 MiB



Toggle details

Dataset Pre-Processing

In [4]:
```
1 # View column data types
2 df.dtypes
```

Out[4]:
```
iso_code                                  object
continent                                 object
location                                  object
date                                      object
total_cases                              float64
                                          ...
human_development_index                  float64
excess_mortality_cumulative_absolute     float64
excess_mortality_cumulative              float64
excess_mortality                         float64
excess_mortality_cumulative_per_million  float64
Length: 65, dtype: object
```

In [29]:
```
1 # convert complete dataset to float
2 df =df.astype(float)
```

In [5]:
```
1 # Check Duplicates
2 df.duplicated().sum()
```

Out[5]:
```
0
```

Dataset Pre-Processing

In [6]:
```
# Get statistical data about each column
df.describe(include="all")
```

Out[6]:

	iso_code	continent	location	date	total_cases	new_cases	new_cases_smoothed	total_deaths	new_deaths	new_deaths_smoothed	...	fer
count	132644	124170	132644	132644	1.254100e+05	125408.000000	124365.000000	1.142930e+05	114489.000000	124365.000000	...	
unique	237	6	237	681	NaN	NaN	NaN	NaN	NaN	NaN	...	
top	ARG	Africa	Mexico	2021-06-24	NaN	NaN	NaN	NaN	NaN	NaN	...	
freq	681	32809	681	234	NaN	NaN	NaN	NaN	NaN	NaN	...	
mean	NaN	NaN	NaN	NaN	2.038985e+06	8353.112888	8370.887531	5.045250e+04	184.004114	168.641989	...	
std	NaN	NaN	NaN	NaN	1.163910e+07	43547.802492	43035.659063	2.589636e+05	871.591559	817.796127	...	
min	NaN	NaN	NaN	NaN	1.000000e+00	-74347.000000	-6223.000000	1.000000e+00	-1918.000000	-232.143000	...	
25%	NaN	NaN	NaN	NaN	2.388000e+03	3.000000	10.286000	8.000000e+01	0.000000	0.143000	...	
50%	NaN	NaN	NaN	NaN	2.714850e+04	104.000000	130.143000	7.410000e+02	2.000000	2.000000	...	
75%	NaN	NaN	NaN	NaN	2.641010e+05	1077.000000	1133.429000	6.468000e+03	22.000000	18.571000	...	
max	NaN	NaN	NaN	NaN	2.519418e+08	907963.000000	826457.571000	5.079990e+06	18007.000000	14703.286000	...	

11 rows × 65 columns

Dataset Pre-Processing

In [19]:
```
#count NaN
df.isna().sum()
```

Out[19]:

total_cases	7234
new_cases	7236
new_cases_smoothed	8279
total_deaths	18351
new_deaths	18155
new_deaths_smoothed	8279
total_cases_per_million	7878
new_cases_per_million	7880
new_cases_smoothed_per_million	8918
total_deaths_per_million	18982
new_deaths_per_million	18786
new_deaths_smoothed_per_million	8918
reproduction_rate	28218
icu_patients	116697
icu_patients_per_million	116697
hosp_patients	113952
hosp_patients_per_million	113952
weekly_icu_admissions	131304
weekly_icu_admissions_per_million	131304
weekly_hosp_admissions	130518
weekly_hosp_admissions_per_million	130518
new_tests	77554
total_tests	77299
total_tests_per_thousand	77299
new_tests_per_thousand	77554
new_tests_smoothed	65751
new_tests_smoothed_per_thousand	65751
positive_rate	69775
tests_per_case	70435

In [20]:
```
# replace NaN with 0
df =df.fillna(0)

```

In [21]:
```
#count NaN
df.isna().sum()
```

Out[21]:

total_cases	0
new_cases	0
new_cases_smoothed	0
total_deaths	0
new_deaths	0
new_deaths_smoothed	0
total_cases_per_million	0
new_cases_per_million	0
new_cases_smoothed_per_million	0
total_deaths_per_million	0
new_deaths_per_million	0
new_deaths_smoothed_per_million	0
reproduction_rate	0
icu_patients	0
icu_patients_per_million	0
hosp_patients	0
hosp_patients_per_million	0
weekly_icu_admissions	0
weekly_icu_admissions_per_million	0

Dataset Pre-Processing

In [22]:
```
# checking the min and max values
df['new_deaths'].max()
```

Out[22]: 18007.0

In [24]:
```
# min value
df['new_deaths'].min()
```

Out[24]: -1918.0

In [26]:
```
# total number of negative values
sum(n < 0 for n in df['new_deaths'].values.
```

Out[26]: 124

In [27]:
```
# replace negative values to 0
df['new_deaths']=df['new_deaths'].clip(lower=0)
```

In [28]:
```
# again check total number of negative values
sum(n < 0 for n in df['new_deaths'].values.flatten())
```

Out[28]: 0

In [33]:
```
# remove column
df=df.drop('new_deaths', axis=1)
df.shape
```

Out[33]: (132644, 60)

Dataset Pre-Processing

In [31]:
```
# handle NaN and infinite values
df = df.reset_index()
```

In [12]:
```
# we don't need this column
df=df.drop('date', axis=1)
df.shape
```

Out[12]: (132644, 64)

In [13]:
```
# remove this colum also
df=df.drop('location', axis=1)
df.shape
```

Out[13]: (132644, 63)

In [14]:
```
# remove this colum also
df=df.drop('continent', axis=1)
df.shape
```

Out[14]: (132644, 62)

In [15]:
```
# remove this colum also
df=df.drop('iso_code', axis=1)
df.shape
```

Out[15]: (132644, 61)

In [16]:
```
# remove this colum also
df=df.drop('tests_units', axis=1)
df.shape
```

Out[16]: (132644, 60)

Dataset Pre-Processing

In [39]:
```
# split to 80% for training and 20% for testing
mydataset_train, mydataset_test, target_col_train, target_col_test=train_test_split(mydataset,target_col, test_size=0.20,
```

In [40]:
```
#print into of spliting data
print('Training Features Shape:', mydataset_train.shape)
print('Training Labels Shape:', target_col_train.shape)
print('Testing Features Shape:', mydataset_test.shape)
print('Testing Labels Shape:', target_col_test.shape)
print('Dataset Shape:', mydataset.shape)
```

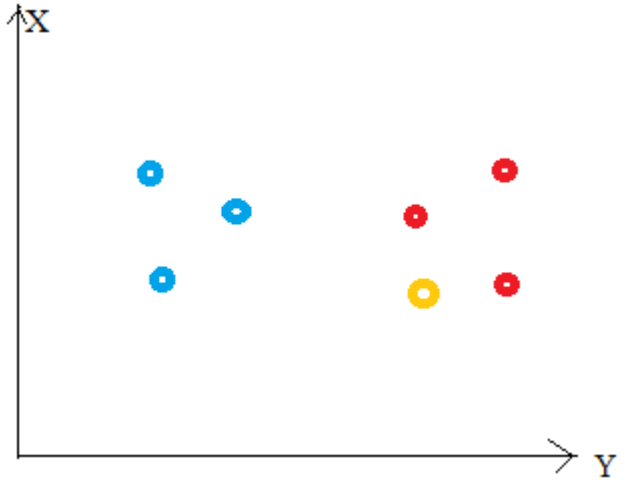
```
Training Features Shape: (106115, 61)
Training Labels Shape: (106115,)
Testing Features Shape: (26529, 61)
Testing Labels Shape: (26529,)
Dataset Shape: (132644, 61)
```

Algorithm

The method K Nearest Neighbor (KNN) is simple to learn and implement.

The K Nearest Neighbor method is a type of supervised learning technique that is used for classification and regression.

It's critical to understand how to pick K value and distance measure.



Algorithm Implementation

In [ ]:
```
#.............Fit Classifier....#
nNNAC = KNeighborsClassifier(n_neighbors=20)
nNNAC.fit(mydataset_train, target_col_train)

```

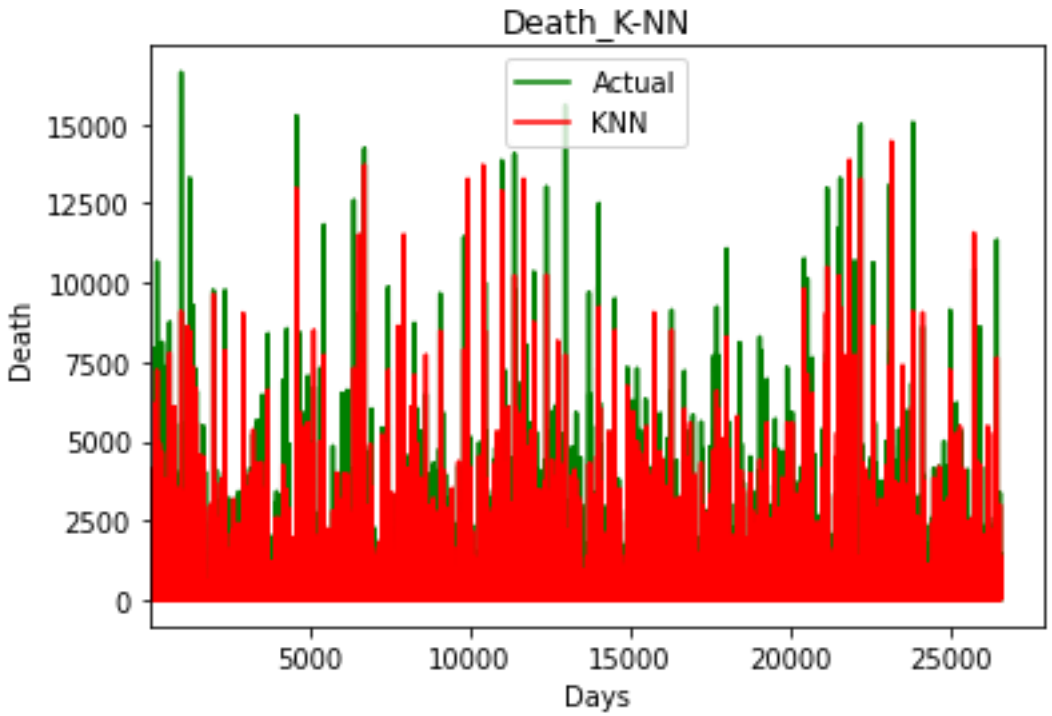
In [ ]:
```
# predict at test data
predictions1 = nNNAC.predict(mydataset_test)
```

In [ ]:
```
# Plot of load forcasted by K-NN Classifier
plt.figure()
plt.ylabel('Death');
plt.xlabel('Days');
plt.plot(target_col_test,'-g',label='Actual');
plt.plot(predictions1,'-r',label='KNN')
plt.title('Death_K-NN')
plt.gca().set_xlim(left=1); plt.legend(fancybox=True, framealpha=0.5)
plt.gca().legend(('Actual','KNN'));
plt.savefig('DeathForecastingKNN.png',bbox_inches='tight',transparent='true')
plt.show();
```

Algorithm Results

In [ ]:
```
# evaluation matrices
print(metrics.accuracy_score(target_col_test, predictions1))
```

```
0.4849787025519243
```



Results

Dataset contains 65 columns and 132644 records

There are 44.0% cells with missing values

Several columns contain NaN values

Some columns contain negative values which is human error

New case and death rate of age more than 65 is comparatively high

K Nearest Neighbor algorithm can efficiently predict the new cases and new death before occurrence

Government can take the appropriate decision before occurrence

Limitations

Dataset does not contain the information about recovered cases

Dataset contains several columns which don’t make sense for this project

Several columns contain NaN values

New_deaths columns contain negative values which mean there is human error which should be considered

Due to very large dataset, machine learning algorithms can’t train accurately

Tools

- **Numpy:** a library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- **Pandas:** a library offers data structures and operations for manipulating numerical tables and time series.
- **Pandas_Profiling:** an open source Python library with which we can quickly do an exploratory data analysis with just a few lines of code.
- **Matplotlib:** a plotting library for the Python programming language and its numerical mathematics extension NumPy
- **Sklearn:** Scikit-learn is a free software machine learning library for the Python programming language.

Thank You