COVID-19 Progression Prediction

1] Importing the required packages and csv file:-

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
patient = pd.read_csv('D:/class/M.Tech 2nd/DL/lab/projects/coronavirusdataset/patient.csv')
patient.head(5)
```

Out[1]:

	patient_id	sex	birth_year	country	region	disease	group	infection_reason	infection_o
0	1	female	1984.0	China	filtered at airport	NaN	NaN	visit to Wuhan	
1	2	male	1964.0	Korea	filtered at airport	NaN	NaN	visit to Wuhan	
2	3	male	1966.0	Korea	capital area	NaN	NaN	visit to Wuhan	
3	4	male	1964.0	Korea	capital area	NaN	NaN	visit to Wuhan	
4	5	male	1987.0	Korea	capital area	NaN	NaN	visit to Wuhan	

In [2]:

```
patient.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7869 entries, 0 to 7868
Data columns (total 15 columns):
patient_id          7869 non-null int64
sex                 679 non-null object
birth_year          666 non-null float64
country             7869 non-null object
region              437 non-null object
disease             28 non-null float64
group               86 non-null object
infection_reason    154 non-null object
infection_order     36 non-null float64
infected_by         70 non-null float64
contact_number      53 non-null float64
confirmed_date      7869 non-null object
released_date       56 non-null object
deceased_date       36 non-null object
state               7869 non-null object
dtypes: float64(5), int64(1), object(9)
memory usage: 645.5+ KB
```

In [4]:

```
patient['state'].value_counts()
```

Out[4]:

```
isolated    1839
released     879
deceased      53
Name: state, dtype: int64
```

2] Data pre-processing:-

Adding new feature age by subtracting current year with birth year feature.

In [5]:

```
patient['age'] = 2020 - patient['birth_year']
```

In [6]:

```
deceased = patient.loc[patient['state'] == 'deceased']
released = patient.loc[patient['state'] == 'released']
isolated = patient.loc[patient['state'] == 'isolated']
```

In [7]:

```
#Adding one more feature to deceased dataset which will contain the number of days patient
date_column = ["confirmed_date","deceased_date"]
for i in date_column:
    deceased[i] = pd.to_datetime(deceased[i])
deceased["no_of_days_survived"] = deceased["deceased_date"] - deceased["confirmed_date"]
deceased.head(5)
```

Out[7]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease
504	1100000071	NaN	male	1941.0	79.0	Korea	Busan	Busanjin-gu	NaN
528	1100000095	NaN	female	1932.0	88.0	Korea	Busan	etc	NaN
530	1100000097	NaN	male	1947.0	73.0	Korea	Busan	Busanjin-gu	NaN
555	1200000038	38.0	female	1963.0	57.0	Korea	Daegu	Nam-gu	True
594	1200000114	114.0	male	1946.0	74.0	Korea	Daegu	NaN	NaN

In [8]:

```
#Adding one more feature to deceased dataset which will contain the number of days patient
date_column = ["confirmed_date","released_date"]
for i in date_column:
    released[i] = pd.to_datetime(released[i])
released["no_of_days_treated"] = released["released_date"] - released["confirmed_date"]
released.head(5)
```

Out[8]:

	patient_id	global_num	sex	birth_year	age	country	province	city	disease	in
0	1000000001	2.0	male	1964.0	56.0	Korea	Seoul	Gangseo-gu	NaN	o
1	1000000002	5.0	male	1987.0	33.0	Korea	Seoul	Jungnang-gu	NaN	o
2	1000000003	6.0	male	1964.0	56.0	Korea	Seoul	Jongno-gu	NaN	
3	1000000004	7.0	male	1991.0	29.0	Korea	Seoul	Mapo-gu	NaN	o
4	1000000005	9.0	female	1992.0	28.0	Korea	Seoul	Seongbuk-gu	NaN	

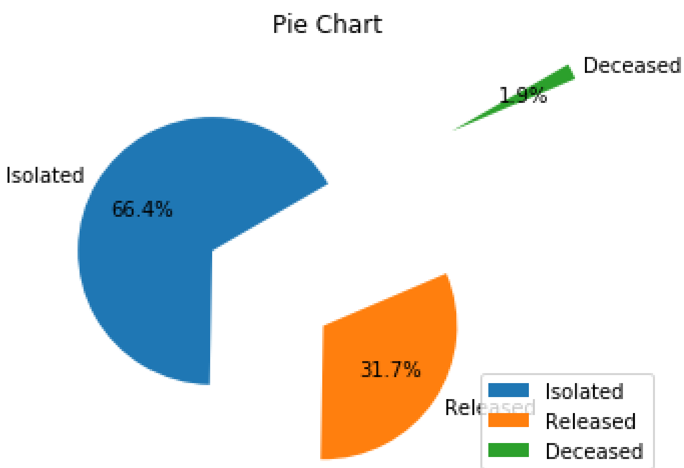
In [9]:

```
print('The percentage of released patient is: ',(len(released) * 100) / len(patient))
print('The percentage of deceased patient is: ',(len(deceased) * 100) / len(patient))
print('The percentage of isolated patient is: ',(len(isolated) * 100) / len(patient))
```

```
The percentage of released patient is:  31.72140021652833
The percentage of deceased patient is:  1.912666907253699
The percentage of isolated patient is:  66.36593287621797
```

In [10]:

```
state = 'Isolated', 'Released', 'Deceased'
sizes = [(len(isolated) * 100) / len(patient),(len(released) * 100) / len(patient),(len(dec
explode = (0,1,2)
fig, ax = plt.subplots()
ax.pie(sizes, explode=explode, labels=state, autopct='%.1f%%',
        shadow=False, startangle=30)
ax.axis('equal')
plt.legend()
plt.title('Pie Chart')
plt.show()
```



The above pie chart shows that around 98.9% of total patient is under isolation whereas around 0.7% patient got discharged and unfortunately 0.4% patient couldn't survived.

3] Linear Regression:-

Step 1:
Computing total number of cases for each confirmed date.

In [3]:

```
#Calculating total number of confirmed cases for each day
case_count_per_day = patient.groupby('confirmed_date').patient_id.count()
case_count_per_day = pd.DataFrame(case_count_per_day)
```

Step 2:
Computing the cumulative sum of case for each date.

In [4]:

```
#Calculating cumulative sum of confirmed cases as date increased(total number of cases incr
data = case_count_per_day.cumsum()
#Picking up the continuous data w.r.t. dates
dataset = data.iloc[16:]
```

Step 3:
Selecting the range of dates and total number of future date that want to be predicted.

In [5]:

```
# This var will be used to predict the cases till next 7 days
days_in_future = 7
dates = pd.date_range('2020-2-20','2020-3-11')

#This is to predict the cases for future dates
future_y_pred = np.array([i for i in range(len(dates)+days_in_future)]).reshape(-1, 1)

#This var will be used to compute the R^2
y_pred = np.array([i for i in range(len(dates))]).reshape(-1, 1)
```

Step 4:
Re-shaping the data to fit it in our model.

In [6]:

```
#Re-shaping the data
x = np.array([i for i in range(len(dates))]).reshape(-1, 1) # index -> ndarray
y = np.array(dataset).reshape(-1, 1) # count->ndarray
```

Step 5:
Fitting the model and predicting the output.

In [7]:

```
from sklearn.linear_model import LinearRegression

linear_model = LinearRegression()
linear_model.fit(x, y)
linear_pred = linear_model.predict(future_y_pred)
```

Step 6:
Calculating coefficient of determination(R^2).

In [8]:

```
y_pred = linear_model.predict(y_pred)
r_sq = linear_model.score(x,y)
print("The coefficient of determination(R^2) for this model is: "+"{:.2f}".format(r_sq*100)
```

```
The coefficient of determination(R^2) for this model is: 96.91 %
```

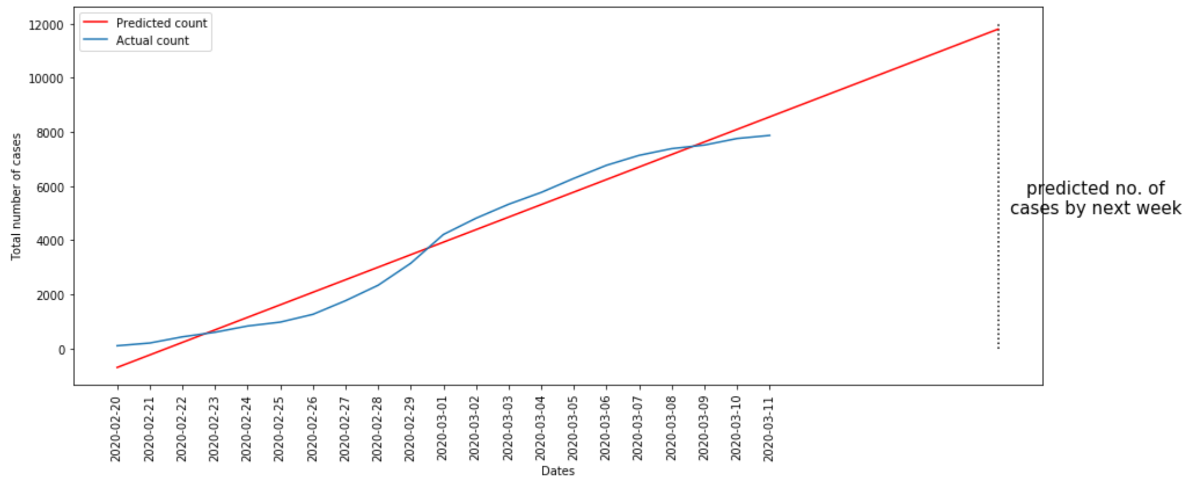
- Coefficient of determination of 96.91% shows that more than 96% of the data fit our linear regression model.
- Generally, a higher coefficient indicates a better fit for the model.

Step 7:

Plotting the graph with confirmed date in X-axes and linear model predicted and actual number of case in Y-axes.

In [9]:

```
#Size of graph
plt.figure(figsize=(15,6))

#Plotting linear model predicted number case for each date(curent + future dates)
plt.plot(linear_pred, color='red', label='Predicted count')

#Plotting actual number of cases for each date
plt.plot(dataset, label='Actual count')

#Labeling X and Y axes.
plt.xlabel('Dates')
plt.ylabel('Total number of cases')

#Drawing a vertical line which touches linear model predicted last value
plt.vlines(x=len(linear_pred)-1, ymin=0, ymax=12000, linestyles='dotted')
plt.text(x=len(linear_pred)+2, y=5000, s='predicted no. of\ncases by next week',color='blac
         fontsize =15,horizontalalignment='center')
plt.xticks(rotation=90)

plt.legend()
plt.show()
```



Observation:-

- By observing the rate of change of total number of cases as date changes, we've predicted the expected total number of cases for next week(i.e. 7th day).
- The predicted total number of cases for next week(i.e. on 18th March) is approximately 11900.