UNIVERSITI TEKNOLOGI MARA

Cawangan Melaka
Kampus Jasin

# MINI PROJECT DATACAMP SIRI - 5

## BY GROUP: KFHM

| STUDENT NAME | MATRIC ID | CODE PROGRAM |
|---|---|---|
| MUHAMMAD KHAIRUL AZHAR BIN KASAN | 2019336691 | CS230 |
| NUR FARHANAH AMIRAH BINTI MISDAN | 2018297966 | CS230 |
| NUR HAZIRAH BINTI MOHD RASID | 2018802092 | CS230 |
| SITI MARIAM BINTI MOHD SALLEH | 2018435902 | CS230 |

DataXpert @UiTM Siri-5 (Final Project)

Refer to the provided data in .csv files and answer the following questions.

1. **List top 3 state that most infected with Covid19**

**Step 1:** Load all the .csv files and do some transformation for states rows in death_cases("terrenganu" to "terengganu") to make sure the data is correctly spelled.

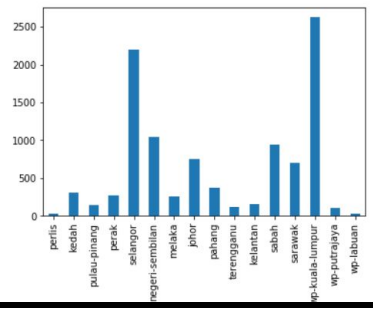
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn.linear_model import LinearRegression

        cv_19  = pd.read_csv('covid-19-malaysia.csv')
        cv_death = pd.read_csv('covid-19-my-death-cases.csv')
        state_case = pd.read_csv('covid-19-my-states-cases.csv')
        cv_death['states']=cv_death['states'].replace('terrenganu','terengganu')
        cv_death
```

Out[1]:

| | no | case | nationality | gender | age | history | chronic | treatment-date | death-date | hospital | states |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 178 | MYS | m | 34 | tabligh | NaN | 12/3/2020 | 17/3/2020 | permai | johor |
| 1 | 2 | 358 | MYS | m | 60 | NaN | y | 14/3/2020 | 17/3/2020 | general | sarawak |
| 2 | 3 | 152 | MYS | m | 58 | tabligh | NaN | NaN | 20/3/2020 | tawau | sabah |
| 3 | 4 | 238 | MYS | m | 50 | tabligh | NaN | 12/3/2020 | 21/3/2020 | melaka | melaka |
| 4 | 5 | 1031 | MYS | f | 79 | NaN | NaN | 16/3/2020 | 18/3/2020 | borneo-medical-centre | sarawak |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 123 | 124 | 8743 | MYR | f | 63 | NaN | y | 9/7/2020 | 26/7/2020 | general | sarawak |
| 124 | 125 | 8974 | PHL | m | 64 | NaN | y | - | 29/7/2020 | queen-elizabeth | sabah |
| 125 | 126 | 9290 | MYS | m | 75 | NaN | y | 24/8/2020 | 29/8/2020 | sultanah-bahiyah | kedah |
| 126 | 127 | 9124 | MYS | m | 62 | tawar-cluster | y | 13/8/2020 | 30/8/2020 | sultanah-bahiyah | kedah |
| 127 | 128 | 9174 | MYS | f | 80 | tawar-cluster | y | 15/8/2020 | 1/9/2020 | pulau-pinang | penang |

**Step 2:** For listing the state, we need to look at the tables that are needed. First, we dropped any null rows in the table. Then, we identified wp-putrajaya as an object column. We convert wp-putrajaya to dtype int64 using astype method, but there's a '-' in wp-putrajaya, and we need to replace that '-' with 0 then we can convert it to type int64. We use `.max()` to find the maximum number of cases in each state then we use `.nlargest()` function to compute the top 3 states with the largest number of cases. The results obtained are WP Kuala Lumpur with the highest value followed by Selangor and Negeri Sembilan.

```
In [137]: state_case.dropna(inplace=True)
          state_case['wp-putrajaya'] = state_case['wp-putrajaya'].replace('-', 0)
          state_case['wp-putrajaya'] = state_case['wp-putrajaya'].astype('float')
          s = state_case.set_index('date')
          summ = s.max()
          print(summ.nlargest(3))
          summ.plot.bar()

          wp-kuala-lumpur    2625.0
          selangor           2199.0
          negeri-sembilan    1043.0
          dtype: float64

Out[137]: <matplotlib.axes._subplots.AxesSubplot at 0x193bef6a448>
```



**2. Malaysia is being divided into 5 regions as follows. Draw a graph of cases, discharged and death according to region. (5 graphs will be produced)**

**Step 1:** We combined the two tables which are cv_19 and state_case using the merge function to view the correlation between these tables and remove the missing values. Then, we replaced the '-' string in wp-putrajaya column with 0 to change the data type to `float.`

```
In [3]: comb_19 = cv_19.merge(state_case, on='date').dropna()
        comb_19['wp-putrajaya'] = comb_19['wp-putrajaya'].replace('-',0)
        comb_19['wp-putrajaya'] = comb_19['wp-putrajaya'].astype('float')
        comb_19
```

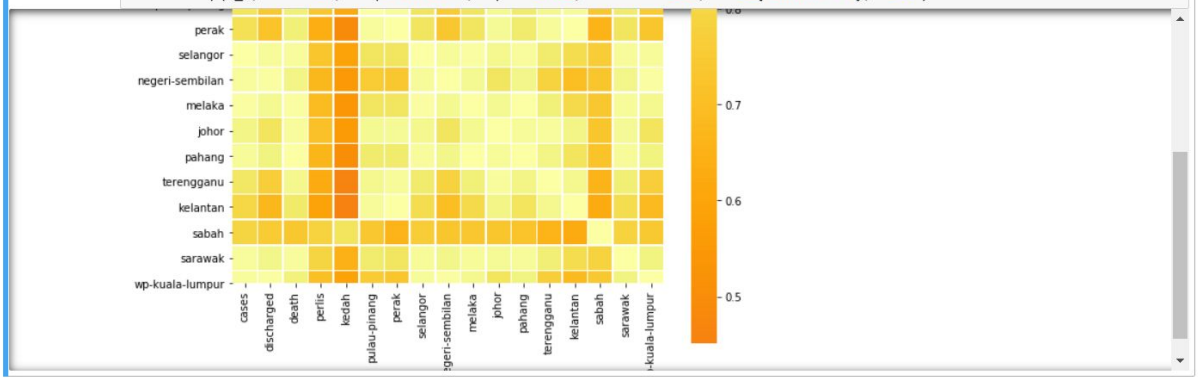
Out[3]:

| | date | cases | discharged | death | icu | perlis | kedah | pulau-pinang | perak | selangor | ... | melaka | johor | pahang | terengganu | kelantan | sabah | sarawak |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 13/3/2020 | 197 | 32 | 0 | 4 | 1.0 | 5.0 | 7.0 | 2.0 | 87.0 | ... | 1.0 | 20.0 | 2.0 | 0.0 | 3.0 | 15.0 | 0.0 |
| 1 | 14/3/2020 | 238 | 35 | 0 | 5 | 2.0 | 5.0 | 7.0 | 2.0 | 92.0 | ... | 6.0 | 22.0 | 2.0 | 0.0 | 3.0 | 26.0 | 6.0 |
| 2 | 16/3/2020 | 553 | 42 | 0 | 12 | 8.0 | 31.0 | 15.0 | 18.0 | 144.0 | ... | 14.0 | 52.0 | 19.0 | 4.0 | 18.0 | 57.0 | 21.0 |
| 3 | 17/3/2020 | 673 | 49 | 2 | 12 | 8.0 | 31.0 | 23.0 | 23.0 | 161.0 | ... | 17.0 | 77.0 | 28.0 | 7.0 | 25.0 | 82.0 | 29.0 |
| 4 | 18/3/2020 | 790 | 60 | 2 | 15 | 8.0 | 36.0 | 30.0 | 28.0 | 192.0 | ... | 18.0 | 88.0 | 29.0 | 10.0 | 30.0 | 103.0 | 49.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 182 | 12/9/2020 | 9868 | 9189 | 128 | 9 | 33.0 | 289.0 | 138.0 | 269.0 | 2193.0 | ... | 262.0 | 752.0 | 371.0 | 114.0 | 160.0 | 808.0 | 700.0 |
| 183 | 13/9/2020 | 9915 | 9196 | 128 | 9 | 33.0 | 303.0 | 138.0 | 269.0 | 2194.0 | ... | 262.0 | 752.0 | 371.0 | 114.0 | 160.0 | 839.0 | 700.0 |
| 184 | 14/9/2020 | 9946 | 9203 | 128 | 11 | 33.0 | 304.0 | 138.0 | 269.0 | 2194.0 | ... | 262.0 | 752.0 | 371.0 | 114.0 | 160.0 | 864.0 | 700.0 |
| 185 | 15/9/2020 | 9969 | 9209 | 128 | 14 | 33.0 | 304.0 | 138.0 | 269.0 | 2199.0 | ... | 262.0 | 752.0 | 371.0 | 114.0 | 160.0 | 877.0 | 700.0 |

**Step 2:** We dropped the columns that are not needed which are date, icu, wp-putrajaya and wp-labuan and assign it to comb_19. Then, we compute the correlation of comb_19 and assign in to n_r to plot graph using `sns.heatmap()`

```
In [4]: comb_19 = comb_19.drop(columns=['date','icu','wp-putrajaya','wp-labuan'])
```

```
In [140]: n_r=comb_19.corr()

          fig, ax = plt.subplots(figsize=(9,9))
          sns.heatmap(n_r, center=0, cmap='inferno', square=True, linewidths=.5, cbar={"shrink":.5}, ax=ax)
```



**Step 3:** To find the number of death cases for each state. Using the fillna method to fill in any null rows. Then create a pivot table which states become the index and use aggregate function to get the size. Next, convert the pivot table into a dataframe called death_cv and reset the index. Later, we rename the column from 0 to 'death'.

In [141]:
```
death = cv_death.fillna(method='bfill')
d = []
dups = death.pivot_table(index = 'states', aggfunc ='size')
dups

death_cv = pd.DataFrame(dups)
death_cv = death_cv.reset_index()
death_cv = death_cv.rename(columns={0:'death'})
death_cv
```

Out[141]:

| | states | death |
|---|---|---|
| 0 | johor | 22 |
| 1 | kedah | 3 |
| 2 | kelantan | 3 |
| 3 | kuala-lumpur | 27 |
| 4 | melaka | 5 |
| 5 | negeri-sembilan | 8 |
| 6 | pahang | 7 |
| 7 | penang | 2 |
| 8 | perak | 6 |
| 9 | perlis | 1 |
| 10 | sabah | 8 |
| 11 | sarawak | 19 |
| 12 | selangor | 16 |
| 13 | terengganu | 1 |

**Step 4:** We use table state_case to find total cases in each state and drop any missing values in the table and define a new empty list called state to find total cases for each state. Then, we convert the list to a dataframe and rename the column name to ‘states’ and ‘total_cases’. After that, we rename ‘wp-kuala-lumpur’ and ‘pulau-pinang’ to ‘kuala-lumpur’ and ‘penang’ respectively. This is because we are following the name given in the question.

In [142]:
```
state_case = state_case.dropna()
state = []
for i in state_case.columns:
    state.append([i,state_case[i].max()])
statess = pd.DataFrame(state)
statess=statess.rename(columns={0:'states',1:'total_cases'})
statess['states'] = statess['states'].replace('wp-kuala-lumpur','kuala-lumpur')
statess['states'] = statess['states'].replace('pulau-pinang','penang')
statess.drop(statess.index[0], inplace=True)
statess
```

Out[142]:

| | states | total_cases |
|---|---|---|
| 1 | perlis | 33 |
| 2 | kedah | 305 |
| 3 | penang | 139 |
| 4 | perak | 269 |
| 5 | selangor | 2199 |
| 6 | negeri-sembilan | 1043 |
| 7 | melaka | 262 |
| 8 | johor | 752 |
| 9 | pahang | 371 |
| 10 | terengganu | 114 |
| 11 | kelantan | 160 |
| 12 | sabah | 937 |
| 13 | sarawak | 700 |
| 14 | kuala-lumpur | 2625 |
| 15 | wp-putrajaya | 99 |

**Step 5:** After getting the total cases for both death and states tables, now it's time to merge these two tables and assign into a variable called dah_merge. Then, we change the data type for total_cases columns to int64 and then reset the index for dah_merge and assign it to a variable named final. Lastly, we split the states according to 5 different regions (north,east,central,south,east Malaysia).

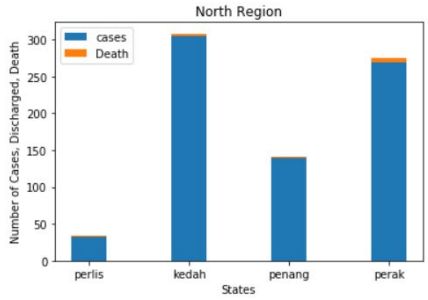
```
In [8]: dah_merge = statess.merge(death_cv, on='states')
        dah_merge['total_cases'] = dah_merge['total_cases'].astype('int64')
        dah_merge = dah_merge.drop_duplicates('states')
        final = dah_merge.reset_index(drop=True)
        north_r = final[final['states'].isin(['perlis','penang','kedah','perak'])]
        east_r = final[final['states'].isin(['pahang','terengganu','kelantan'])]
        central_r = final[final['states'].isin(['selangor','kuala-lumpur'])]
        south_r = final[final['states'].isin(['negeri-sembilan','melaka','johor'])]
        east_m = final[final['states'].isin(['sabah','sarawak'])]
```

• **Northern Region**: Perlis, Kedah, Penang, Perak.

We plot the north region by using a stacked barplot and plotting by states column, total_cases, and death.

```
#plotting time
fig,ax = plt.subplots()

#north_region
ax.bar(north_r.states, north_r.total_cases, width=0.35, label='cases')
ax.bar(north_r.states, north_r.death, width=0.35, bottom=north_r.total_cases, label='Death')
ax.legend()
ax.set_xlabel('States')
ax.set_ylabel('Number of Cases, Discharged, Death')
ax.set_title('North Region')
```

Out[8]: Text(0.5, 1.0, 'North Region')

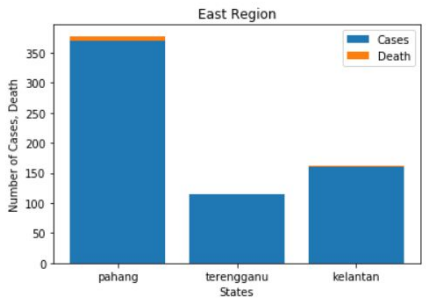



• **East Coast Region**: Kelantan, Terengganu, Pahang.

We plot the east coast region by using a stacked barplot and plotting by states column, total_cases, and death.

In [9]:
```
east_r.reset_index(inplace=True)
fig,ax2 = plt.subplots()
ax2.bar(east_r.states, east_r.total_cases, label='Cases')
ax2.bar(east_r.states, east_r.death,bottom=east_r.total_cases, label='Death')
ax2.legend()
ax2.set_xlabel('States')
ax2.set_ylabel('Number of Cases, Death')
ax2.set_title('East Region')
```

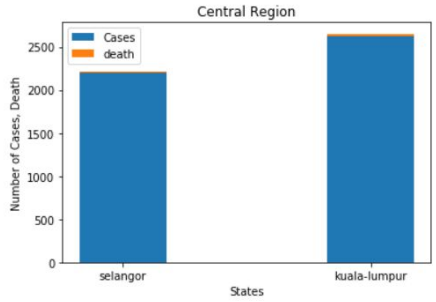
Out[9]: Text(0.5, 1.0, 'East Region')



• **Central Region**: Selangor, Kuala Lumpur.

We plot the central region by using a stacked barplot and plotting by states column, total_cases, and death.

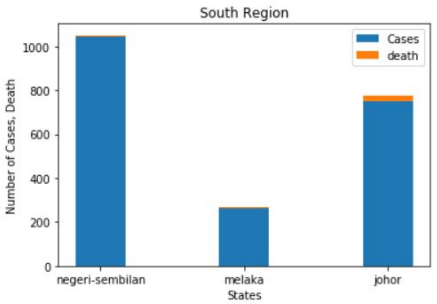
```
In [10]: central_r.reset_index(inplace=True)
         fig, ax3 = plt.subplots()
         ax3.bar(central_r.states, central_r.total_cases, width=0.35, label='Cases')
         ax3.bar(central_r.states, central_r.death, bottom=central_r.total_cases, width=0.35, label='death')
         ax3.legend()
         ax3.set_xlabel('States')
         ax3.set_ylabel('Number of Cases, Death')
         ax3.set_title('Central Region')

Out[10]: Text(0.5, 1.0, 'Central Region')
```




• **Southern Region**: Negeri Sembilan, Malacca, Johor.

We plot the southern region by using a stacked barplot and plotting by states column, total_cases, and death.

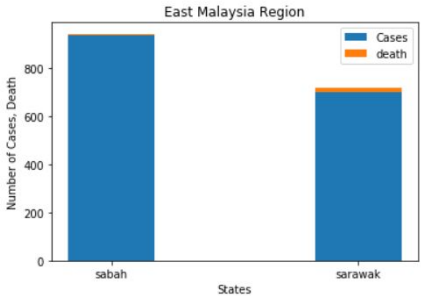
```
In [11]: south_r.reset_index(inplace=True)
         fig, ax4 = plt.subplots()
         ax4.bar(south_r.states, south_r.total_cases, width=0.35, label='Cases')
         ax4.bar(south_r.states, south_r.death, bottom=south_r.total_cases, width=0.35, label='death')
         ax4.legend()
         ax4.set_xlabel('States')
         ax4.set_ylabel('Number of Cases, Death')
         ax4.set_title('South Region')

Out[11]: Text(0.5, 1.0, 'South Region')
```



• **East Malaysia**: Sabah Sarawak

We plot east Malaysia by using a stacked barplot and plotting by states column, total_cases, and death.

```
In [12]: east_m.reset_index(inplace=True)
         fig, ax5 = plt.subplots()
         ax5.bar(east_m.states, east_m.total_cases, width=0.35, label='Cases')
         ax5.bar(east_m.states, east_m.death, bottom=east_m.total_cases, width=0.35, label='death')
         ax5.legend()
         ax5.set_xlabel('States')
         ax5.set_ylabel('Number of Cases, Death')
         ax5.set_title('East Malaysia Region')

Out[12]: Text(0.5, 1.0, 'East Malaysia Region')
```




**3. Based on news, most infected patients are male. Verify the statement together with the group of age of the patient infected.**

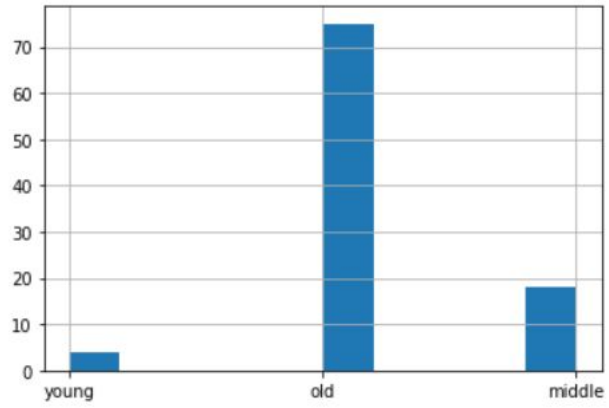
We use table cv_death and filter the gender column to get the male data. We fill the null value using fillna method with (`method = 'bfill'`) `as` the argument and assign it to variable `m`. We also use `np.nan` function to replace `'??'` string in the age column. After dropping missing values in `m`, we convert age datatype to int64 to filter their range age according to the three categories which are young, middle, and old. Using the `.loc()` function to compute each category and assign it to Age_group, we then print the values according to the categories and plot the result obtained into a histogram.

```
In [14]: male = cv_death[cv_death.gender == 'm']
         m = male.fillna(method='bfill')
         m['age']=m['age'].replace('??',np.nan)
         m = m.dropna()
         m['age']=m['age'].astype('int64')
         young = (m['age']>18) & (m['age']<=35)
         middle = (m['age']>35) & (m['age']<=55)
         old = m['age']>55
         m.loc[young,'Age_group']='young'
         m.loc[middle,'Age_group']='middle'
         m.loc[old, 'Age_group']='old'

         print(m.Age_group.value_counts())
         print('total death: ',len(cv_death))
         m.Age_group.hist()

         old       75
         middle    18
         young      4
         Name: Age_group, dtype: int64
         total death:  128
```

This is the result obtained. As we can see the total for death cases is 128, and the total for death cases for the male is 97 which is more than half of the other gender.

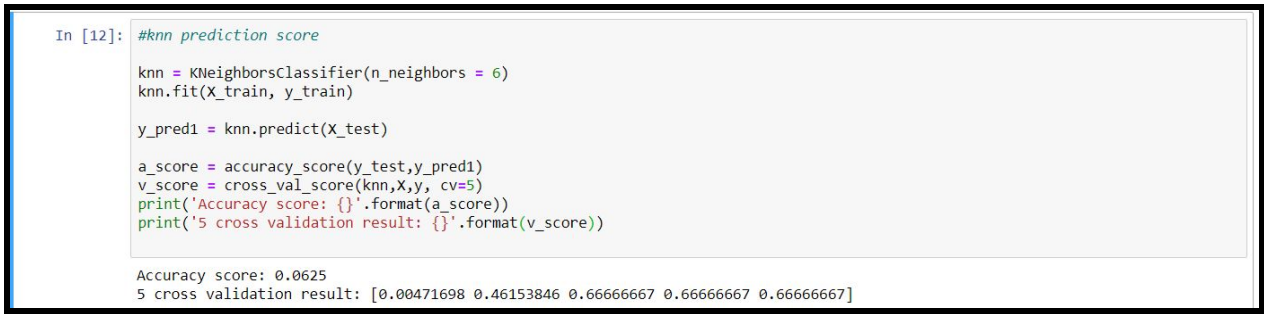
```
old       75
middle    18
young      4
Name: Age_group, dtype: int64
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1f9a402ddc8>



**+ Develop and test your own hypotheses/questions to understand more about the characteristics of the Covid19 wave in Malaysia.**

KNN is not the best algorithm to be used for finding characteristics of Covid-19 waves in Malaysia for the current dataset acquired. More data needed to get a better model for doing prediction.



In [12]:
```
#knn prediction score

knn = KNeighborsClassifier(n_neighbors = 6)
knn.fit(X_train, y_train)

y_pred1 = knn.predict(X_test)

a_score = accuracy_score(y_test,y_pred1)
v_score = cross_val_score(knn,X,y, cv=5)
print('Accuracy score: {}'.format(a_score))
print('5 cross validation result: {}'.format(v_score))
```

```
Accuracy score: 0.0625
5 cross validation result: [0.00471698 0.46153846 0.66666667 0.66666667 0.66666667]
```

**+Develop a machine learning model to predict the numbers of infected by Covid19 in Malaysia for any particular day based on the previous day's information.**

**Step 1:** Import the required package, then import the data from a csv file called covid-19-malaysia.csv, then inspect the data.

In [4]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error,accuracy_score
from sklearn.neighbors import KNeighborsClassifier

df = pd.read_csv('covid-19-malaysia.csv', parse_dates=['date'])
df['date'] = pd.to_datetime(df['date'])
#df['year'] = df['date'].map(lambda x: x.year)
#df['month'] = df['date'].map(lambda x: x.month)
df.info()
df.describe()
df.corr()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 237 entries, 0 to 236
Data columns (total 5 columns):
date          237 non-null datetime64[ns]
cases         237 non-null int64
discharged    237 non-null int64
death         237 non-null int64
icu           237 non-null int64
dtypes: datetime64[ns](1), int64(4)
memory usage: 9.4 KB
```

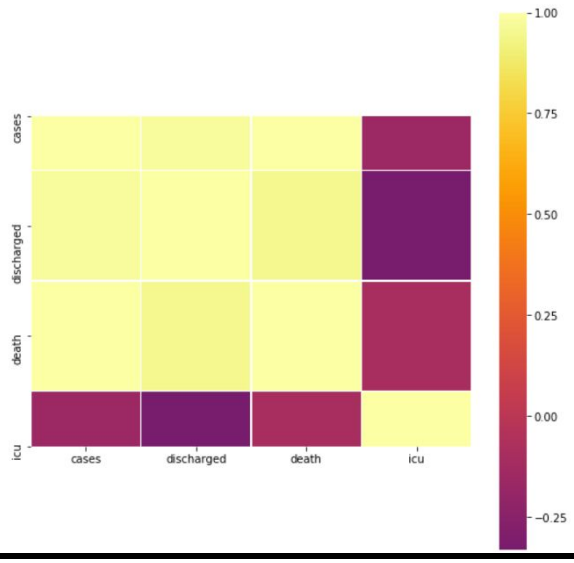
Out[17]:

| | cases | discharged | death | icu |
|---|---|---|---|---|
| cases | 1.000000 | 0.975645 | 0.989894 | -0.150486 |
| discharged | 0.975645 | 1.000000 | 0.945745 | -0.330656 |
| death | 0.989894 | 0.945745 | 1.000000 | -0.097992 |
| icu | -0.150486 | -0.330656 | -0.097992 | 1.000000 |

**Step 2:** Plotting a heatmap for correlation of the columns using seaborn.

```
In [18]: import seaborn as sns

         fig, ax = plt.subplots(figsize=(9,9))
         sns.heatmap(df.corr(), center=0, cmap='inferno', square=True, linewidths=.5, cbar={"shrink":.5}, ax=ax)
```

```
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x19d6f553a08>
```




**Step 3:** Select the data for X and y, then by using train_test_split function, we split the data into a ratio of 80:20. Next, we initialized the LinearRegression function as lr and fit the data X_train and y_train. Then, we predict by using the X_test data and assign it to variable y_pred. We print out the score, root mean squared error(rmse), interception slope, and we print some prediction done by using our own data.

```
In [28]: X = df.drop(['cases','date'], axis=1)
         y = df.cases
         #y = y.ravel()
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42, shuffle=True)

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

         y_pred = lr.predict(X_test)

         print("R^2: {}".format(lr.score(X_test, y_test)))
         rmse = np.sqrt(mean_squared_error(y_test,y_pred))
         print("Root Mean Squared Error: {}".format(rmse))
         print('Coefficients: ',lr.coef_)
         print('Interception:',lr.intercept_)
         print('predicted new cases: ',np.round(lr.predict([[2034,120,9]])))
```

```
R^2: 0.9977656874096448
Root Mean Squared Error: 177.54844701615167
Coefficients:  [ 0.54244699 33.76523166 11.76691665]
Interception: 108.46657226549723
predicted new cases:  [5370.]
```