Bush Fire Data:

Data Source:

Fire History Records of Fires primarily on Public Land showing the fire scars
https://discover.data.vic.gov.au/dataset/designated-bushfire-prone-area-bpa
(https://discover.data.vic.gov.au/dataset/designated-bushfire-prone-area-bpa)(08/09/2011 to 31/12/2020)

Have two files

- postcode file
- shape file
  - Name of place
  - coordinates of that place (boundary coordinates)

First need to get the codes for the only victoria

that might would be easier to search

In [ ]:

```
### importing the libraries

import pandas as pd
import os
```

In [ ]:

```
# os.getcwd()
```

Postcode file

In [ ]:

```
### postcode file:

pc = pd.read_csv("postcode.csv")
pc.head(2)
```

- removing the unwanted columns and rows as we want the data for all states of the victoria

In [ ]:

```
newpc = pc.loc[(pc['admin code1'] == 'VIC')&(pc['postal code'] < 8000),
               ['postal code','place name','admin name1', 'latitude', 'longitude']].reset_i
```

In [ ]:

```
### lets save the file

newpc.to_csv("postcode_vic.csv", index=False)
```

In [ ]:

```
## import the postcode file

pc_vic = pd.read_csv("postcode_vic.csv")
pc_vic.head(2)
```

Bushfire Shape file:

shape file extraction is in two files:

- location name and other data
- coordinations of area burnt

In [ ]:

```
!pip install pyshp
import shapefile
import pandas as pd
import os

data = shapefile.Reader(r'C:\Users\JxD\Desktop\IE\SDM841179\ll_gda94\sde_shape\whole\VIC\FI

# data.fields

corrd = [i.points for i in data.shapes()]


index=[]
lat=[]
long=[]

for i,j in enumerate(corrd):
    idx = i
    for a,b in j:
        index.append(i)
        lat.append(a)
        long.append(b)

DF = pd.DataFrame({'index':index, 'lat':lat,'long':long})
DF.to_csv("corrdinates.csv")



col = data.fields[1:]
col = [ i[0] for i in col]

d_list = [i for i in data.records() ]


import pandas as pd

data = pd.DataFrame(d_list,columns=col )

data.to_csv("data_from_shp.csv",index=False)
```

shape file

In [ ]:

```
### shape file
shape = pd.read_csv('data_from_shp.csv')
shape.head(3)
```

In [ ]:

```
shape.groupby(['SEASON','FIRE_SVRTY']).count()
```

In [ ]:

```
shape['FIRE_SVRTY'].unique()
```

In [ ]:

```
shape['index']=shape.index
```

In [ ]:

```
#### take the imp columns and make a new csv file
### as data is big so will take the bushfire data from year 2008 to 2020


df = shape.loc[shape['SEASON'] >= 2008,['index','SEASON',"NAME","AREA_HA"]]

df.to_csv("shape_2k8_20.csv", index=False)
```

In [ ]:

```
shape_2k8_20 = pd.read_csv("shape_2k8_20.csv")
print(shape_2k8_20.shape)
shape_2k8_20.head()
```

Can see that Name is not proper format according to the requirement

In [ ]:

```
### checking missing values

len(shape_2k8_20.loc[shape_2k8_20['NAME'].isna()])  ### where name value is missing
```

In [ ]:

```
len(shape_2k8_20.loc[shape_2k8_20['NAME'].notna()]) ### where the name given, have to refor
```

So where the name is given we will reformat it and find the postcode with the help of postcode file and make new columns of these values.

In [ ]:

```
## making column for postcode and  place name
shape_2k8_20['Place'] = ''
shape_2k8_20['postcode'] = 0
print(shape_2k8_20.columns)
shape_2k8_20.head(2)
```

In [ ]:

```
## if name not find in data then will add the "NA".

for each in shape_2k8_20.loc[shape_2k8_20['NAME'].notna()].index:
    if (each % 10000) == 0:
        print(each)
    place = str(shape_2k8_20.iloc[each,list(shape_2k8_20.columns).index('NAME')]).split("-"

    shape_2k8_20.iloc[each, list(shape_2k8_20.columns).index('Place')] = place

    if len(pc_vic[pc_vic['place name'] == place]) !=0:
        shape_2k8_20.iloc[each, list(shape_2k8_20.columns).index('postcode')] = pc_vic.iloc

    else:
        shape_2k8_20.iloc[each, list(shape_2k8_20.columns).index('postcode')] = "NA"
        shape_2k8_20.iloc[each, list(shape_2k8_20.columns).index('Place')] = "NA"
        pass
```

In [ ]:

```
shape_2k8_20.to_csv("shape2k8NA.csv", index=False)
```

In [ ]:

```
shape_2k8_20.head()
```

Now the missing Name and 'NA' are still present in data. Will sort that out later.

coordinates file

In [ ]:

```
## extract the coordinates file
```

In [ ]:

```
### read the saved coordinates file

corr = pd.read_csv('corrdinates.csv')
```

In [ ]:

```
corr.head(2)
```

In [ ]:

```
#### index of places with 'NA'   +  no name in shape file

na = list(shape_2k8_20[shape_2k8_20['Place'] == 'NA']['index'])

nName_na = list(shape_2k8_20.loc[shape_2k8_20['NAME'].isna()]['index']) + na

len(nName_na)
```

In [ ]:

```
# ### find all of the indexes at which we don't have nameand 'NA' with NO coordinates.

mis_cor = []
for i in nName_na:
    if len(corr[corr['index'] == i]) ==0:
        mis_cor.append(i)
```

In [ ]:

```
pd.DataFrame(mis_cor, columns=['mis_cor']).to_csv("mis_corr.csv", index=False)  ## save the
```

In [ ]:

```
mis_cor = pd.read_csv('mis_corr.csv')
```

In [ ]:

```
mis_cor.head(2)
```

In [ ]:

```
### NO name

mc = mis_cor['mis_cor'].values

shape_2k8_20[shape_2k8_20['index'] == mc[0]]
```

In [ ]:

```
## No coordinates
corr[corr['index'] == mc[0]]
```

In [ ]:

```
len(mc)  ### these are the number of data which dont have the name as well as the coordinat
```

In [ ]:

```
## remain data which have corrdinates but no have name. will compute the name
len(nName_na) - len(mc)
```

In [ ]:

```
### index at which name is not present but coordinates are given.

x = []
for e in nName_na:
    if (e not in mc):
        x.append(e)
```

In [ ]:

```
## x contains all of the rows which have no name but have corrdinates. we will compute thei
## coordinates


def check(i):

    lon = corr.loc[corr['index']==i]['longitude'].mean()
    lat = corr.loc[corr['index']==i]['latitude'].mean()

    bw = 0.005

    length = len(pc_vic.loc[(pc_vic.latitude > lat-bw)& (pc_vic.latitude < lat+bw) & (pc_vi

    while length != 1:

        if length >= 2:
            idx = pc_vic.loc[(pc_vic.latitude > lat-bw)& (pc_vic.latitude < lat+bw) & (pc_v
            shape_2k8_20.loc[shape_2k8_20['index'] == i, 'postcode'] = int(pc_vic.iloc[idx,
            shape_2k8_20.loc[shape_2k8_20['index'] == i, 'Place'] = pc_vic.iloc[idx,list(pc
            return


        bw +=0.005
        length = len(pc_vic.loc[(pc_vic.latitude > lat-bw)& (pc_vic.latitude < lat+bw) & (p

    ix = pc_vic.loc[(pc_vic.latitude > lat-bw)& (pc_vic.latitude < lat+bw) & (pc_vic.longit

    shape_2k8_20.loc[shape_2k8_20['index'] == i, 'postcode'] = int(pc_vic.iloc[ix,list(pc_v
    shape_2k8_20.loc[shape_2k8_20['index'] == i, 'Place'] = pc_vic.iloc[ix,list(pc_vic.colu
```

In [ ]:

```
shape_2k8_20[shape_2k8_20['index'] == x[1]]  ## no name
```

In [ ]:

```
corr[corr['index'] == x[1]]  ### but have coordinates
```

In [ ]:

```
for i in x:
    check(i)
```

In [ ]:

```
shape_2k8_20.to_csv("final_shape_2k8.csv", index=False)
```

In [ ]:

```
# shape_2k8_20 = pd.read_csv("final_shape_2k8.csv")
```

In [ ]:

```
## removing those data which don't have coordinates and 'NA' with no coordinates

indx_list = []
for i in x:
    indx_list.append(shape_2k8_20[shape_2k8_20['index'] == i].index[0])
```

In [ ]:

```
shape_2k8_20.iloc[indx_list].to_csv('clean_shape_2k8.csv', index = False)
```

In [ ]:

```
shape_2k8_20 = pd.read_csv('clean_shape_2k8.csv')
```

In [ ]:

```
shape_2k8_20.head()
```