```
#### Collating required datapoints, reshaping them and calculating important features ###

import xlrd
import pandas as pd
import numpy as np
import os
from time import strptime
import datetime

def get_arterial(file_path,category):
    """
    variable path is the path of the xls workbook and category is "rural" / "urban" / "all",
    returns dataframe containing the values for given category for each state
    """
    book = xlrd.open_workbook(file_path)
    file_name = os.path.basename(file_path)
    year = str(20) + "".join([str(s) for s in file_name if s.isdigit()]) ## gets the year from
filename
    Month = strptime(file_name[2:5],'%b').tm_mon ## gets month no
    mydate = datetime.date(int(year),Month, 1) ## first day of the month and year
    mydate_1 = mydate - datetime.timedelta(days=1) ## interested in last month of this year
as data corresponds to last month and same year
    mydate_2 = mydate - datetime.timedelta(days=368) ## interested in last month of last
year as data corresponds to last month and last year
    #monthid1 = str(mydate_1.strftime("%Y")) + str(mydate_1.strftime("%m")) ## 200706 for
July 2007 file
    monthid2 = str(mydate_2.strftime("%Y")) + str(mydate_2.strftime("%m")) ## 200606 for
July 2007 file
    try:
        if category.lower() == "rural":
            index = 3
        elif category.lower() == "urban":
            index = 4
        else:
            index = 5
        sheet = book.sheet_by_index(index)
        list_states = sheet.col_values(0)
        xstart = list_states.index('Connecticut')
        xend = list_states.index('TOTALS')
        #list1 = sheet.col_slice(colx= 8,start_rowx=xstart,end_rowx= xend - 1)
        #list1 = [w.value for w in list1]
        list2 = sheet.col_slice(colx= 9,start_rowx=xstart,end_rowx= xend - 1)
        list2 = [w.value for w in list2]
        list3 = sheet.col_slice(colx= 0,start_rowx=xstart,end_rowx= xend - 1)
        list3 = [w.value.lower() for w in list3] ## take lowercase for direct match later
        df = pd.concat([pd.DataFrame(list3),pd.DataFrame(list2)], axis = 1) #
,pd.DataFrame(list1)
```

```
        #col_name_1 = category + '_Arterial_' + monthid1
        col_name_2 = category + '_Arterial_' + monthid2
        df.columns = ['State', col_name_2 ] # col_name_1,
        df[col_name_2].replace('', np.nan, inplace=True) ## removes rows with blank records (
zonal categories)
        df['State'].replace('', np.nan, inplace=True)
        curr_monthid = str(mydate.strftime("%Y")) + str(mydate.strftime("%m")) ## 200707 for
July 2007 file
        df['data_monthid'] = curr_monthid
        df.dropna(subset=[col_name_2], inplace=True)
        df.dropna(subset=['State'], inplace=True)
        df = df[~df.State.str.contains("subtotal")] ### causes problems on joins, there in most
files
        df = df[df.State != "total"] ## causes problems on joins, is there only in specific files
        df['State'] = df.State.str.strip() ## removes leading and lagging white spaces if any
        df2 = pd.melt(df,id_vars=['State','data_monthid'],var_name=['category'],
value_name='Million_Vehicle_Miles')
        return df2
    except:
        print("error in file ",os.path.basename(file_path))

## get all the files
def filelist(root):
    """Return a fully-qualified list of filenames under root directory"""
    allfiles = []
    for path, subdirs, files in os.walk(root):
        for name in files:
            if name.find("xls") >= 0:
                allfiles.append(os.path.join(path, name))
    return allfiles

file_list = filelist('/Users/MrMndFkr/Desktop/linear-regression-project/Datasets')


### check function get_arterial and append dataframes for Dataset 1
for file in file_list:
    try:
        df1 = get_arterial(file,"Rural")
        df2 = get_arterial(file,"Urban")
        df3 = get_arterial(file,"All")
        df_final = pd.concat([df1,df2,df3], axis = 0)
        #df_temp = pd.merge(df1,df2, how = 'inner', on = 'State')
        #df_final = pd.merge(df_temp,df3, how = 'inner', on = 'State')
        #assert df_final.shape[0] == df3.shape[0]
        #assert df_final.shape[0] == df_temp.shape[0]
    except:
        print('error encountered at ' + os.path.basename(file))
```

```
## get view of large no of columns and rows
pd.set_option('display.max_rows', 500)
pd.set_option('display.max_columns', 500)
pd.set_option('display.width', 1000)

## appending these dataframes
df1 = get_arterial(file_list[0],"Rural")
df2 = get_arterial(file_list[0],"Urban")
df3 = get_arterial(file_list[0],"All")
df_final = pd.concat([df1,df2,df3], axis = 0)
#df_temp = pd.merge(df1,df2, how = 'inner', on = 'State')
#df_final = pd.merge(df_temp,df3, how = 'inner', on = 'State')
for file in file_list[1:]:
   try:
      df1 = get_arterial(file,"Rural")
      df2 = get_arterial(file,"Urban")
      df3 = get_arterial(file,"All")
      df_final = pd.concat([df_final,df1,df2,df3], axis = 0)
      #df_temp = pd.merge(df1,df2, how = 'inner', on = 'State')
      #df_temp2 = pd.merge(df_temp,df3, how = 'inner', on = 'State')
      #df_final = pd.merge(df_final,df_temp2, how = 'inner', on = 'State')
      #assert df_final.shape[0] == df_temp.shape[0]
      #assert df_final.shape[0] == df_temp2.shape[0]
   except:
      print('error encountered at ' + os.path.basename(file))


## only change is that the required datapoints are in cols 7 and 8 for Dataset III ( in Dataset
I, they are in cols 9 and 10)
def get_arterial(file_path,category):
   """
   variable path is the path of the xls workbook and category is "rural" / "urban" / "all",
   returns dataframe containing the values for given category for each state
   """
   book = xlrd.open_workbook(file_path)
   file_name = os.path.basename(file_path)
   year = str(20) + "".join([str(s) for s in file_name if s.isdigit()]) ## gets the year from
filename
   Month = strptime(file_name[2:5],'%b').tm_mon ## gets month no
   mydate = datetime.date(int(year),Month, 1) ## first day of the month and year
   #mydate_1 = mydate - datetime.timedelta(days=1) ## interested in last month of this
year as data corresponds to last month and same year
   mydate_2 = mydate - datetime.timedelta(days=368) ## interested in last month of last
year as data corresponds to last month and last year
   #monthid1 = str(mydate_1.strftime("%Y")) + str(mydate_1.strftime("%m")) ## 200706 for
July 2007 file
```

```
   monthid2 = str(mydate_2.strftime("%Y")) + str(mydate_2.strftime("%m")) ## 200606 for
July 2007 file
   try:
      if category.lower() == "rural":
         index = 3
      elif category.lower() == "urban":
         index = 4
      else:
         index = 5
      sheet = book.sheet_by_index(index)
      list_states = sheet.col_values(0)
      xstart = list_states.index('Connecticut')
      xend = list_states.index('TOTALS')
      #list1 = sheet.col_slice(colx= 6,start_rowx=xstart,end_rowx= xend - 1)
      #list1 = [w.value for w in list1]
      list2 = sheet.col_slice(colx= 7,start_rowx=xstart,end_rowx= xend - 1)
      list2 = [w.value for w in list2]
      list3 = sheet.col_slice(colx= 0,start_rowx=xstart,end_rowx= xend - 1)
      list3 = [w.value.lower() for w in list3] ## take lowercase for direct match later
      df = pd.concat([pd.DataFrame(list3),pd.DataFrame(list2)], axis = 1) #
pd.DataFrame(list1),
      #col_name_1 = category + '_Arterial_' + monthid1
      col_name_2 = category + '_Arterial_' + monthid2
      df.columns = ['State',  col_name_2 ] ## col_name_1,
      df[col_name_2].replace('', np.nan, inplace=True) ## removes rows with blank records (
zonal categories)
      df['State'].replace('', np.nan, inplace=True)
      curr_monthid = str(mydate.strftime("%Y")) + str(mydate.strftime("%m")) ## 200707 for
July 2007 file
      df['data_monthid'] = curr_monthid
      df.dropna(subset=[col_name_2], inplace=True)
      df.dropna(subset=['State'], inplace=True)
      df = df[~df.State.str.contains("subtotal")] ### causes problems on joins, there in most
files
      df = df[df.State != "total"] ## causes problems on joins, is there only in specific files
      df['State'] = df.State.str.strip() ## removes leading and lagging white spaces if any
      df2 = pd.melt(df,id_vars=['State','data_monthid'],var_name=['category'],
value_name='Million_Vehicle_Miles')
      return df2
   except:
      print("error in file ",os.path.basename(file_path))

# get new filelist from dataset III
file_list = filelist('/Users/MrMndFkr/Desktop/linear-regression-project/Datasets_III')

## get collated dataset
#df1 = get_arterial(file_list[0],"Rural")
```

```
#df2 = get_arterial(file_list[0],"Urban")
#df3 = get_arterial(file,"All")
#df_temp = pd.merge(df1,df2, how = 'inner', on = 'State')
#df_final = pd.merge(df_temp,df3, how = 'inner', on = 'State')
for file in file_list[1:]:
    try:
        df1 = get_arterial(file,"Rural")
        df2 = get_arterial(file,"Urban")
        df3 = get_arterial(file,"All")
        df_final = pd.concat([df_final,df1,df2,df3], axis = 0)
        #df_temp = pd.merge(df1,df2, how = 'inner', on = 'State')
        #df_temp2 = pd.merge(df_temp,df3, how = 'inner', on = 'State')
        #df_final = pd.merge(df_final,df_temp2, how = 'inner', on = 'State')
        #assert df_final.shape[0] == df_temp.shape[0]
        #assert df_final.shape[0] == df_temp2.shape[0]
    except:
        print('error encountered at ' + os.path.basename(file))


## Now fetching the 2018 data for months after Jun 2018
def get_arterial(file_path,category):
    """
    variable path is the path of the xls workbook and category is "rural" / "urban" / "all",
    returns dataframe containing the values for given category for each state
    """
    book = xlrd.open_workbook(file_path)
    file_name = os.path.basename(file_path)
    year = str(20) + "".join([str(s) for s in file_name if s.isdigit()]) ## gets the year from
filename
    Month = strptime(file_name[2:5],'%b').tm_mon ## gets month no
    mydate = datetime.date(int(year),Month, 1) ## first day of the month and year
    mydate_1 = mydate - datetime.timedelta(days=1) ## interested in last month of this year
as data corresponds to last month and same year
    #mydate_2 = mydate - datetime.timedelta(days=368) ## interested in last month of last
year as data corresponds to last month and last year
    monthid1 = str(mydate_1.strftime("%Y")) + str(mydate_1.strftime("%m")) ## 200706 for
July 2007 file
    #monthid2 = str(mydate_2.strftime("%Y")) + str(mydate_2.strftime("%m")) ## 200606 for
July 2007 file
    try:
        if category.lower() == "rural":
            index = 3
        elif category.lower() == "urban":
            index = 4
        else:
            index = 5
        sheet = book.sheet_by_index(index)
```

```
        list_states = sheet.col_values(0)
        xstart = list_states.index('Connecticut')
        xend = list_states.index('TOTALS')
        #list1 = sheet.col_slice(colx= 6,start_rowx=xstart,end_rowx= xend - 1)
        #list1 = [w.value for w in list1]
        list2 = sheet.col_slice(colx= 8,start_rowx=xstart,end_rowx= xend - 1)
        list2 = [w.value for w in list2]
        list3 = sheet.col_slice(colx= 0,start_rowx=xstart,end_rowx= xend - 1)
        list3 = [w.value.lower() for w in list3] ## take lowercase for direct match later
        df = pd.concat([pd.DataFrame(list3),pd.DataFrame(list2)], axis = 1) #
pd.DataFrame(list1),
        #col_name_1 = category + '_Arterial_' + monthid1
        col_name_2 = category + '_Arterial_' + monthid1
        df.columns = ['State',  col_name_2 ] ## col_name_1,
        df[col_name_2].replace('', np.nan, inplace=True) ## removes rows with blank records (
zonal categories)
        df['State'].replace('', np.nan, inplace=True)
        curr_monthid = str(mydate.strftime("%Y")) + str(mydate.strftime("%m")) ## 200707 for
July 2007 file
        df['data_monthid'] = curr_monthid
        df.dropna(subset=[col_name_2], inplace=True)
        df.dropna(subset=['State'], inplace=True)
        df = df[~df.State.str.contains("subtotal")] ### causes problems on joins, there in most
files
        df = df[df.State != "total"] ## causes problems on joins, is there only in specific files
        df['State'] = df.State.str.strip() ## removes leading and lagging white spaces if any
        df2 = pd.melt(df,id_vars=['State','data_monthid'],var_name=['category'],
value_name='Million_Vehicle_Miles')
        return df2
    except:
        print("error in file ",os.path.basename(file_path))


file_list = ['/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/18augtvt.xls', \
             '/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/18septvt.xls', \
             '/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/18octtvt.xls', \
             '/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/18novtvt.xls', \
             '/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/18dectvt.xls', \
             '/Users/MrMndFkr/Desktop/linear-regression-project/Datasets/19jantvt.xls']

## get collated dataset
for file in file_list:
    try:
        df1 = get_arterial(file,"Rural")
        df2 = get_arterial(file,"Urban")
        df3 = get_arterial(file,"All")
        df_final = pd.concat([df_final,df1,df2,df3], axis = 0, sort = True)
```

```
    except:
        print('error encountered at ' + os.path.basename(file))

## basic QC
df_final['monthid'] = df_final.category.str[-6:]
df_final['year'] = df_final.category.str[-6:-2]
df_final.monthid.value_counts(sort = True)
qc = df_final.monthid.value_counts().sort_index()
max(qc), min(qc) ## equal value confirms all monthids have the same no of states
df_final.year.value_counts().sort_index()

# remove data for 2002
df_final['year'] = pd.to_numeric(df_final.year).astype(dtype = 'int64')
df_final.year.value_counts().sort_index()
df_final = df_final[df_final.year >= 2003]
df_final.year.value_counts().sort_index()

## get different column for category
df_final.loc[df_final.category.str.find("Rural") >= 0,'area'] = 'rural'
df_final.loc[df_final.category.str.find("Urban") >= 0,'area'] = 'urban'
df_final.loc[df_final.category.str.find("All") >= 0,'area'] = 'all'
#df_final.iloc[0:10,]

## changing variables into appropriate datatypes
df = df_final.copy()
df['Million_Vehicle_Miles'] = pd.to_numeric(df.Million_Vehicle_Miles, errors='coerce')
df['monthid'] = pd.to_numeric(df.monthid, errors='coerce')

## 12 month rolling sum - overall vehicle miles
df = df.loc[df.area == 'all']
df.sort_values(['State','monthid'], ascending = True, inplace = True)
df['rolling_miles_12m'] =
df.groupby(['State'])['Million_Vehicle_Miles'].rolling(12).sum().values

## sub regions
Northeast = list(map(lambda x:x.lower(), ['Connecticut','Maine','Massachusetts','New
Hampshire','New Jersey','New York','Pennsylvania','Rhode Island','Vermont']))
Southatlantic = list(map(lambda x:x.lower(), ['Delaware','District of
Columbia','Florida','Georgia','Maryland','North Carolina','South Carolina','Virginia','West
Virginia']))
Northcentral = list(map(lambda x:x.lower(),
['Illinois','Indiana','Iowa','Kansas','Michigan','Minnesota','Missouri','Nebraska','North
Dakota','Ohio','South Dakota','Wisconsin']))
Southgulf = list(map(lambda x:x.lower(),
['Alabama','Arkansas','Kentucky','Louisiana','Mississippi','Oklahoma','Tennessee','Texas']))
```

```
West = list(map(lambda x:x.lower(),
['Alaska','Arizona','California','Colorado','Hawaii','Idaho','Montana','Nevada','New
Mexico','Oregon','Utah','Washington','Wyoming']))
df.loc[df.State.isin(Northeast),'region'] = 'North_East'
df.loc[df.State.isin(Southatlantic),'region'] = 'South_Atlantic'
df.loc[df.State.isin(Northcentral),'region'] = 'North_Central'
df.loc[df.State.isin(Southgulf),'region'] = 'South_Gulf'
df.loc[df.State.isin(West),'region'] = 'West'

## make quarter
df['date'] = pd.to_datetime(df.monthid, format='%Y%m')
df['quarter'] = df.date.dt.quarter
df['quarter'] = df.year*10 +  df.quarter
#df.drop(columns = 'quarterid', inplace = True)

## create dataset at a quarterly level
data = df.groupby('quarter').agg({'rolling_miles_12m' : ['mean']}).reset_index()
data.columns = ['quarter','avg_miles_rollingsum_12m']
data = data[data.quarter >= 20041]

## calculate area wise percentages
a = df.groupby(['quarter','region']).agg({'rolling_miles_12m':['sum']}).reset_index()
a.columns = ['quarter','region','total']
a = a.pivot(columns = 'region',index = 'quarter', values = 'total')

b = df.groupby(['quarter']).agg({'rolling_miles_12m':['sum']}).reset_index()
b.columns = ['quarter','val']

c = pd.merge(a, b, how = 'inner', on = 'quarter')
c['perc_north_central'] = c.North_Central / c.val
c['perc_north_east'] = c.North_East / c.val
c['perc_south_atlantic'] = c.South_Atlantic / c.val
c['perc_west'] = c.West / c.val

## get them into dataset
data =
pd.merge(data,c[['quarter','perc_north_central','perc_north_east','perc_south_atlantic','per
c_west']], how = 'inner', on = 'quarter')

# urban vehicular miles
df = df_final.copy()
df['Million_Vehicle_Miles'] = pd.to_numeric(df.Million_Vehicle_Miles, errors='coerce')
df['monthid'] = pd.to_numeric(df.monthid, errors='coerce')
df = df.loc[df.area == 'urban']
df.sort_values(['State','monthid'], ascending = True, inplace = True)
df['rolling_miles_12m'] =
df.groupby(['State'])['Million_Vehicle_Miles'].rolling(12).sum().values
```

```
df['date'] = pd.to_datetime(df.monthid, format='%Y%m')
df['quarter'] = df.date.dt.quarter
df['quarter'] = df.year*10 +  df.quarter
df.sort_values(['State','monthid'], ascending = True, inplace = True)
df['rolling_miles_12m'] =
df.groupby(['State'])['Million_Vehicle_Miles'].rolling(12).sum().values
d = df.groupby('quarter').agg({'rolling_miles_12m' : ['mean']}).reset_index()
d.columns = ['quarter','urban_avg_rolling_miles_12m']
d.head()

## merge urban and get percentage
data = pd.merge(data,d, how = 'inner', on = 'quarter')
data['perc_urban'] = data.urban_avg_rolling_miles_12m / data.avg_miles_rollingsum_12m
data =
data[['quarter','avg_miles_rollingsum_12m','perc_north_central','perc_north_east','perc_so
uth_atlantic','perc_west','perc_urban']]

## save it to disk
data.to_csv("/Users/MrMndFkr/Desktop/linear-regression-project/data_reshaped.csv")
```