# Pre-processing

## Import Libraries and Define functions

In [329]:
```
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

from sklearn.metrics import r2_score
import statsmodels.api as sm
from statsmodels.api import OLS
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.linear_model import RidgeCV
from sklearn.linear_model import LassoCV
from sklearn.linear_model import LinearRegression

from sklearn.model_selection import KFold
from sklearn.decomposition import PCA

#import pydotplus
#import io
from sklearn.tree import export_graphviz
from IPython.display import Image
from IPython.display import display
import seaborn as sns
%matplotlib inline
```

In [330]:
```
def despine():
    sns.despine(left=True, bottom=True)

def get_axs(rows, columns, fig_size_width, fig_size_height):
    dims = (fig_size_width, fig_size_height)
    fig, axs = plt.subplots(rows, columns, figsize=dims)
    if(rows*columns>1):
         axs = axs.ravel()
    return axs

def get_accuracy_model(X, Y, model):
    Y_pred = model.predict(X)
    misclassification_rate = np.mean([int(x) for x in Y_pred != Y])
    return 1 - misclassification_rate

def get_accuracy_pred(Y, Y_pred):
    misclassification_rate = np.mean([int(x) for x in Y_pred != Y])
    return 1 - misclassification_rate

def split_dataset(data, train_size_pc, y_col):
    np.random.seed(9001)
    msk = np.random.rand(len(data)) < train_size_pc
    data_train = data[msk]
    data_test = data[~msk]

    x_train = data_train.iloc[:,0:y_col]
    y_train = data_train.iloc[:,y_col]

    x_test = data_test.iloc[:,0:y_col]
    y_test = data_test.iloc[:,y_col]
    return x_train, y_train, x_test, y_test

def set_title_xlabel_ylabel(ax, title, xlabel, ylabel):
    ax.set_title(title)
    ax.set_xlabel(xlabel)
    ax.set_ylabel(ylabel)
```

In [331]:
```
sns.set(rc={'axes.facecolor':'white', 'figure.facecolor':'white'})
sns.set_style("whitegrid")
sns.set(font_scale=1.3)
```

## Import required datasets

In [332]:
```
murder_data = pd.read_csv('murder_data.csv',  index_col = 0)
census_data = pd.read_csv('census_data.csv',  index_col = 0)
firearm_data = pd.read_csv('firearm_data.csv',  index_col = 0)
```

In [335]:
```
num_rows = murder_data.shape[0]
murder_data['firearms'] = np.zeros(num_rows)
for row_index in range(0,num_rows):
    row = murder_data.iloc[row_index]
    states = row['msa_state'].split(',')
    firearms = [firearm_data.loc[state]['Firearm'] for state in states]
    row_data = list(murder_data.iloc[row_index])
    row_data[-1] = np.mean(firearms)
    murder_data.iloc[row_index] = row_data
```

In [336]:
```
murder_data['firearms'] = murder_data['firearms'].astype(np.int)
```

In [337]:
```
num_rows = census_data.shape[0]
```

## Combine using MSA codes and State codes

In [339]:
```
census_data['firearms'] = np.zeros(num_rows)
census_data['murder_rate'] = np.zeros(num_rows)

for row_index in range(0, num_rows):
    row = census_data.iloc[row_index]
    murder_rate = murder_data[murder_data['code'] == int(row['msa'])][str(int(
row['year']))]
    firearms    = murder_data[murder_data['code'] == int(row['msa'])]['firearm
s']
    if (len(murder_rate)>0) & (len(firearms)>0):
        murder_rate = murder_rate.iloc[0]
        firearms = firearms.iloc[0]
    else:
        murder_rate = -1
        firearms = -1
    row_data = list(row)
    row_data[-1] = float(murder_rate)
    row_data[-2] = int(firearms)
    census_data.iloc[row_index] = row_data
```

## Export to master dataset

In [340]:
```
census_data.to_csv("crime_data.csv")
```