Filter Method

import time  
import tracemalloc  
import timeit  
start\_time = time.time()  
tracemalloc.start()  
import os  
import pandas as pd  
os.chdir('C:/Users/bitel/PycharmProjects/CIND820\_Project/CIND820\_Project')  
crime = pd.read\_csv('cleanedcommunitiescrime.csv', sep=',')  
##One feature, 'OtherperCap' was an object data type but was supposed to be a numeric value. Change feature to numeric value.  
x = crime.drop('ViolentCrimesPerPop', axis = 1)  
y = crime['ViolentCrimesPerPop']  
#Data was split into training (0.7) and test (0.3) data. The following preprocessing procedures and modelling will occur on the training data.  
from sklearn.model\_selection import train\_test\_split  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size= 0.3, random\_state=345)  
##One feature, 'OtherperCap' was an object data type but was supposed to be a numeric value. Change feature to numeric value.  
x\_train['OtherPerCap'] = pd.to\_numeric(x\_train['OtherPerCap'], errors='coerce')  
print(x\_train.dtypes)  
###Converted the missing/null values into column median values.  
x\_train = x\_train.fillna(0)  
y\_train = y\_train.fillna(0)  
x\_train = x\_train.replace(0, x\_train.median())  
y\_train = y\_train.replace(0, y\_train.median())  
#Conducted Yeo and Johnson transformation on the dataset because numerous features were skewed distributions.  
#In order to improve the predictor variables and possibly the model, this transformation was applied.  
from sklearn.preprocessing import PowerTransformer  
pt = PowerTransformer(method='yeo-johnson')  
x\_train = pd.DataFrame(pt.fit\_transform(x\_train), columns=x\_train.columns)  
#The feature selection embedded method Lasso was applied to the preprocessed training dataset.  
#The top 10 features were selected.  
from sklearn.feature\_selection import SelectKBest  
from scipy.stats import spearmanr  
correlation\_matrix, \_ = spearmanr(x\_train, y\_train, axis=0, nan\_policy='propagate', alternative='two-sided' )  
selection = SelectKBest(score\_func=lambda x\_train, y\_train: correlation\_matrix[:-1, -1], k=10)  
selection.fit(x\_train, y\_train)  
newx\_train = selection.transform(x\_train)  
selected\_features = x\_train.columns[selection.get\_support()].tolist()  
print(selected\_features)  
#k-fold cross validation was used to evaluate the Support Vector Regressor (SVR) model against 10 split samples.  
#The average r2 scores was determined as a performance metric.  
from sklearn.model\_selection import KFold  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.svm import SVR  
cv = KFold(n\_splits=10, random\_state=1, shuffle=True)  
model = SVR()  
score = cross\_val\_score(model, newx\_train, y\_train, scoring='neg\_mean\_squared\_error', cv=cv, n\_jobs=-1)  
mean\_score = sum(score)/10  
model.fit(newx\_train, y\_train)  
print(mean\_score)  
newx\_test = selection.transform(x\_test)  
y\_pred = model.predict(newx\_test)  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score  
mse = mean\_squared\_error(y\_test, y\_pred)  
mae = mean\_absolute\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
adj\_r2 = 1 - ((1 - r2) \* (598 - 1) / (598 - 97 - 1))  
print(mse, mae, r2, adj\_r2)  
snapshot = tracemalloc.take\_snapshot()  
end\_time = time.time()  
tracemalloc.stop()  
memory = snapshot.statistics('lineno')  
for stat in memory[:10]:  
 print(stat)  
print(end\_time - start\_time)  
t = timeit.timeit(stmt='''import time  
import tracemalloc  
import timeit  
start\_time = time.time()  
tracemalloc.start()  
import os  
import pandas as pd  
os.chdir('C:/Users/bitel/PycharmProjects/CIND820\_Project/CIND820\_Project')  
crime = pd.read\_csv('cleanedcommunitiescrime.csv', sep=',')  
##One feature, 'OtherperCap' was an object data type but was supposed to be a numeric value. Change feature to numeric value.  
x = crime.drop('ViolentCrimesPerPop', axis = 1)  
y = crime['ViolentCrimesPerPop']  
#Data was split into training (0.7) and test (0.3) data. The following preprocessing procedures and modelling will occur on the training data.  
from sklearn.model\_selection import train\_test\_split  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size= 0.3, random\_state=345)  
##One feature, 'OtherperCap' was an object data type but was supposed to be a numeric value. Change feature to numeric value.  
x\_train['OtherPerCap'] = pd.to\_numeric(x\_train['OtherPerCap'], errors='coerce')  
print(x\_train.dtypes)  
###Converted the missing/null values into column median values.  
x\_train = x\_train.fillna(0)  
y\_train = y\_train.fillna(0)  
x\_train = x\_train.replace(0, x\_train.median())  
y\_train = y\_train.replace(0, y\_train.median())  
#Conducted Yeo and Johnson transformation on the dataset because numerous features were skewed distributions.  
#In order to improve the predictor variables and possibly the model, this transformation was applied.  
from sklearn.preprocessing import PowerTransformer  
pt = PowerTransformer(method='yeo-johnson')  
x\_train = pd.DataFrame(pt.fit\_transform(x\_train), columns=x\_train.columns)  
#The feature selection embedded method Lasso was applied to the preprocessed training dataset.  
#The top 10 features were selected.  
from sklearn.feature\_selection import SelectKBest  
from scipy.stats import spearmanr  
correlation\_matrix, \_ = spearmanr(x\_train, y\_train, axis=0, nan\_policy='propagate', alternative='two-sided' )  
selection = SelectKBest(score\_func=lambda x\_train, y\_train: correlation\_matrix[:-1, -1], k=10)  
selection.fit(x\_train, y\_train)  
newx\_train = selection.transform(x\_train)  
selected\_features = x\_train.columns[selection.get\_support()].tolist()  
print(selected\_features)  
#k-fold cross validation was used to evaluate the Support Vector Regressor (SVR) model against 10 split samples.  
#The average r2 scores was determined as a performance metric.  
from sklearn.model\_selection import KFold  
from sklearn.model\_selection import cross\_val\_score  
from sklearn.svm import SVR  
cv = KFold(n\_splits=10, random\_state=1, shuffle=True)  
model = SVR()  
score = cross\_val\_score(model, newx\_train, y\_train, scoring='neg\_mean\_squared\_error', cv=cv, n\_jobs=-1)  
mean\_score = sum(score)/10  
model.fit(newx\_train, y\_train)  
print(mean\_score)  
newx\_test = selection.transform(x\_test)  
y\_pred = model.predict(newx\_test)  
from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score  
mse = mean\_squared\_error(y\_test, y\_pred)  
mae = mean\_absolute\_error(y\_test, y\_pred)  
r2 = r2\_score(y\_test, y\_pred)  
adj\_r2 = 1 - ((1 - r2) \* (598 - 1) / (598 - 97 - 1))  
print(mse, mae, r2, adj\_r2)''', number=50)  
print(t)