Advanced Statistical Algorithms MA691
Assignment 2

Name: **Naman Goyal**
Roll No: **180123029**

Ques.**1**

Code:

```
import numpy as np
print("------------------Q1----------------\n")

# generating initial normal random sample of size 10 with mean 0, variance
1
initSample = []

for i in range(10):
   initSample.append(np.random.normal(0, 1))

# resampling with replacement and generating new sample
newSample = np.random.choice(initSample, 10)

# Regenerating 1000 more samples for new case
# now verification
# generate 1000 more samples and check the average number of unique
elements from the original sample
uniqueNo = []

for i in range(1000):
   sample = np.random.choice(initSample, 10)
   x = np.unique(sample)
   uniqueNo.append(len(x)/10)

print("For Original Sample")
print("Original Sample: ", initSample)
print("Mean: ", np.mean(initSample))
print("Variance: ", np.var(initSample))
print("---------------------------------\n")
```

```
print("For New Sample")
print("New Sample using non-parametric bootstrap: ", newSample)
print("Mean: ", np.mean(newSample))
print("Variance: ", np.var(newSample))
print("---------------------------------\n")

print("Avg number of elements from original sample: ", np.mean(uniqueNo))
```

Output:

```
↳ python3 180123029_q1.py
-----------------Q1---------------

For Original Sample
Original Sample:  [-1.7946032627518407, 0.3158661086158461, 0.2856272636646265, 0.020712852296235262, 0.2536660301580155, 0.2
79557473167694, -1.036499391474474, -0.15983533667228134, -1.0823853520128492, 0.00012806284115234073]
Mean:  -0.2917765552167876
Variance:  0.4964983869196907
-------------------------------

For New Sample
New Sample using non-parametric bootstrap:  [ 2.53666030e-01 -1.03649939e+00 -1.08238535e+00 -1.79460326e+00
  2.53666030e-01 -1.59835337e-01  1.28062841e-04 -1.08238535e+00
  3.15866109e-01  1.28062841e-04]
Mean:  -0.43322544003101127
Variance:  0.5015216816743528
-------------------------------

Avg number of elements from original sample:  0.6535
```

Ques.**2**

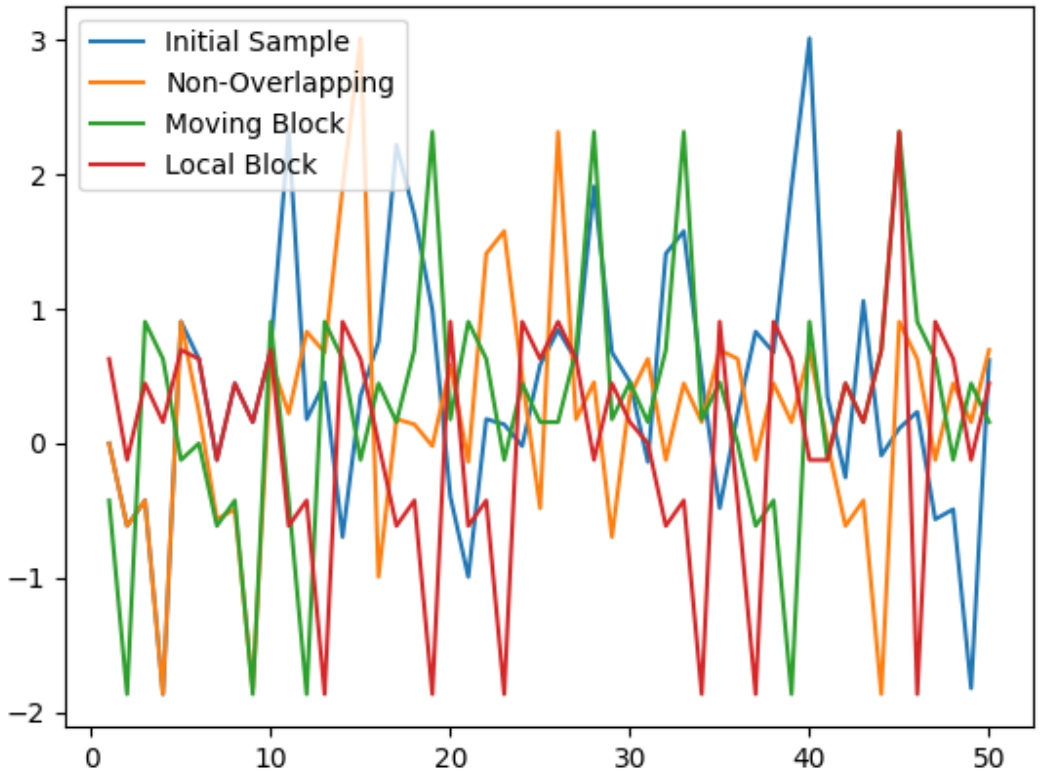
Code:

```
import numpy as np
import matplotlib.pyplot as plt

print("------------------Q2----------------\n")

variance = 1
sigma = np.sqrt(variance)
epsilon = np.random.normal(0, sigma, 50)
```

```
n = 50
x_axis = np.linspace(1, n, n)
initSample = []
initSample.append(0)

for i in range(1, n):
   elem = initSample[i-1]*0.5 + epsilon[i]
   initSample.append(elem)

plt.plot(x_axis, initSample)
l = 5 # Size of Block

# NON-OVERLAPPING
numBlocks = int(n/l)
blocks = np.reshape(initSample, (numBlocks, l))
indexes = [x for x in range(numBlocks)]
resampleBlock = np.random.choice(indexes, numBlocks)
nonOverlap = []

for i in resampleBlock:
   nonOverlap.extend(blocks[i])
plt.plot(x_axis, nonOverlap)

# MOVING BLOCK
numBlocks = n - l + 1
resampleBlock = np.random.choice(indexes, int(n/l))
movBlock = []

for i in resampleBlock:
   movBlock.extend(initSample[i:i+l])
plt.plot(x_axis, movBlock)

# LOCAL BLOCK
numBlocks = n - l + 1
numBlocksReqd = int(n/l)
resampleBlock = []
delt = 4

for i in range(numBlocksReqd):
   # pick first block randomly
```

```
    if i == 0:
        resampleBlock.extend(np.random.choice(indexes, 1))
    else:
        # new block is selected from vicinity of previous block(+/- delt)
        lower_bound = max(0, resampleBlock[i-1] - delt)
        upper_bound = min(numBlocks-1, resampleBlock[i-1] + delt)

resampleBlock.extend(np.random.choice(indexes[lower_bound:upper_bound+1],
1))

localBlock = []

for i in resampleBlock:
    localBlock.extend(initSample[i:i+l])
plt.plot(x_axis, localBlock)

plt.legend(["Initial Sample", "Non-Overlapping", "Moving Block", "Local
Block"])
plt.show()
```

Output:



Ques.**3**

Code:

```
import numpy as np
import math
import matplotlib.pyplot as plt
import pandas as pd
from numpy import loadtxt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import mean_squared_error as mse

print("------------------Q3----------------\n")
```

```
x = [(i+1) for i in range(36)]
y = loadtxt("FRWRD.txt", delimiter="\n", unpack=False)
x = np.array(x)
y = np.array(y)
N = 36

def polyRegressionModel(degree, k_fold):
   poly_features = PolynomialFeatures(degree=degree)
   X_poly = poly_features.fit_transform(x.reshape(-1, 1))
   poly = LinearRegression()
   return np.mean(cross_val_score(poly, X_poly, y.reshape(-1, 1),
cv=k_fold, scoring='neg_mean_squared_error'))

err_Poly3 = polyRegressionModel(3, 10)
print("Error in Polynomial 3")
print(err_Poly3)

err_Poly6 = polyRegressionModel(6, 10)
print("Error in Polynomial 6")
print(err_Poly6)

err_Poly8 = polyRegressionModel(8, 10)
print("Error in Polynomial 8")
print(err_Poly8)

lm = LinearRegression()
lm.fit(x.reshape(-1, 1), y.reshape(-1, 1))
plt.scatter(x, y, s=15, label='Original Pts')

print("Calc for degree 3")
Inp3 = [('polynomial', PolynomialFeatures(degree=3)),('modal',
LinearRegression())]
pipe3 = Pipeline(Inp3)
pipe3.fit(x.reshape(-1, 1), y.reshape(-1, 1))
predPoly3 = pipe3.predict(x.reshape(-1, 1))
sortZip3 = sorted(zip(x, predPoly3))
xPolyfor3, predPoly3 = zip(*sortZip3)
plt.plot(xPolyfor3, predPoly3, label='Polynomial Reg for degree -- 3')

print("Calc for degree 6")
```

```
Inp6 = [('polynomial', PolynomialFeatures(degree=6)),('modal',
LinearRegression())]
pipe6 = Pipeline(Inp6)
pipe6.fit(x.reshape(-1, 1), y.reshape(-1, 1))
predPoly6 = pipe6.predict(x.reshape(-1, 1))
sortZip6 = sorted(zip(x, predPoly6))
xPolyfor6, predPoly6 = zip(*sortZip6)
plt.plot(xPolyfor6, predPoly6, label='Polynomial Reg for degree -- 6')

print("Calc for degree 8")
Inp8 = [('polynomial', PolynomialFeatures(degree=8)),('modal',
LinearRegression())]
pipe8 = Pipeline(Inp8)
pipe8.fit(x.reshape(-1, 1), y.reshape(-1, 1))
predPoly8 = pipe8.predict(x.reshape(-1, 1))
sortZip8 = sorted(zip(x, predPoly8))
xPolyfor8, predPoly8 = zip(*sortZip8)
plt.plot(xPolyfor8, predPoly8, label='Polynomial Reg for degree -- 8')

plt.legend()
plt.show()
```

- Text is been loaded from FRWRD.txt file

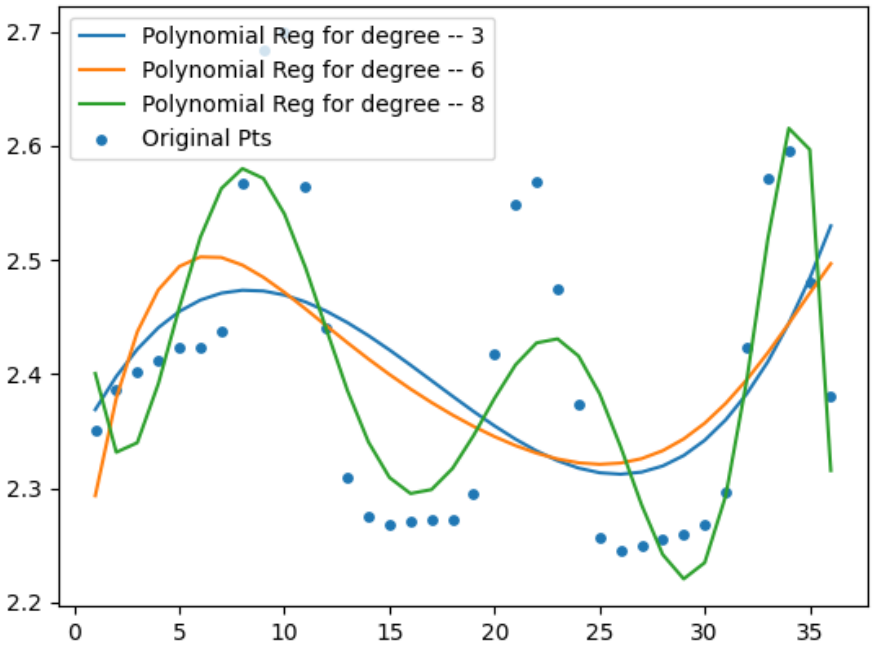
2.351
2.387
2.4020000000000001
2.4119999999999999
2.4239999999999999
2.4239999999999999
2.4369999999999998
2.5670000000000002
2.6840000000000002
2.6989999999999998
2.5640000000000001
2.4409999999999998
2.3090000000000002

2.2759999999999998
2.2679999999999998
2.2709999999999999
2.2730000000000001
2.2730000000000001
2.2949999999999999
2.4180000000000001
2.5489999999999999
2.569
2.4740000000000002
2.3740000000000001
2.2570000000000001
2.2450000000000001
2.25
2.2549999999999999
2.2589999999999999
2.2679999999999998
2.2959999999999998
2.4239999999999999
2.5710000000000002
2.5950000000000002
2.48
2.3799999999999999

Output:
Terminal

```
↳ python3 180123029_q3.py
------------------Q3----------------

Error in Polynomial 3
-0.02304109026676365
Error in Polynomial 6
-1.0964838879763286
Error in Polynomial 8
-4.466656860018206
Calc for degree 3
Calc for degree 6
Calc for degree 8
```

Plot


Polynomial Reg for degree -- 3
Polynomial Reg for degree -- 6
Polynomial Reg for degree -- 8
Original Pts
2.7
2.6
2.5
2.4
2.3
2.2
0
5
10
15
20
25
30
35