PROGRAMMING ASSIGNMENT 2

Submitted by Aaryan Agarwal

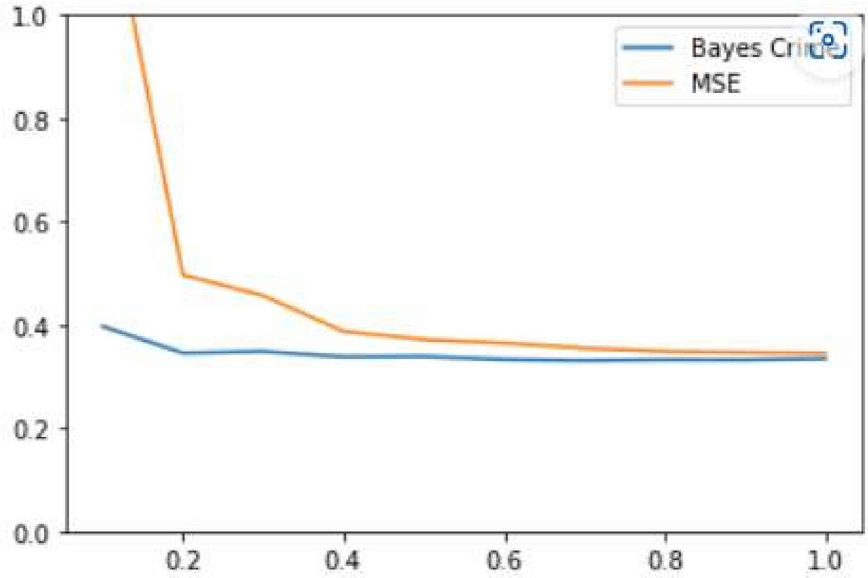
Task 1:

```
{'partition': 0.1, 'alpha': 188.13418564070818, 'beta': 3.0590610997
375354, 'regularization': 61.50063026098103}
{'partition': 0.2, 'alpha': 284.7219942674846, 'beta': 2.96441887394
6527, 'regularization': 96.04647871116627}
{'partition': 0.3, 'alpha': 266.99009988195087, 'beta': 2.8471527729
09647, 'regularization': 93.77441998277473}
{'partition': 0.4, 'alpha': 280.4488295991778, 'beta': 2.84968456413
86455, 'regularization': 98.41399049159224}
{'partition': 0.5, 'alpha': 284.08760024259936, 'beta': 2.9156222181
703813, 'regularization': 97.4363545702676}
{'partition': 0.6, 'alpha': 263.4754615387708, 'beta': 2.96226180404
9953, 'regularization': 88.9440160820869}
{'partition': 0.7, 'alpha': 254.16163967645417, 'beta': 3.0875611127
863833, 'regularization': 82.31793003996117}
{'partition': 0.8, 'alpha': 254.20674738385384, 'beta': 3.1241961399
624962, 'regularization': 81.36708964338757}
{'partition': 0.9, 'alpha': 247.4375126121574, 'beta': 3.04625567703
9333, 'regularization': 81.22677110696198}
{'partition': 1.0, 'alpha': 239.71632213512905, 'beta': 3.0852982512
664147, 'regularization': 77.69632061883588}
```

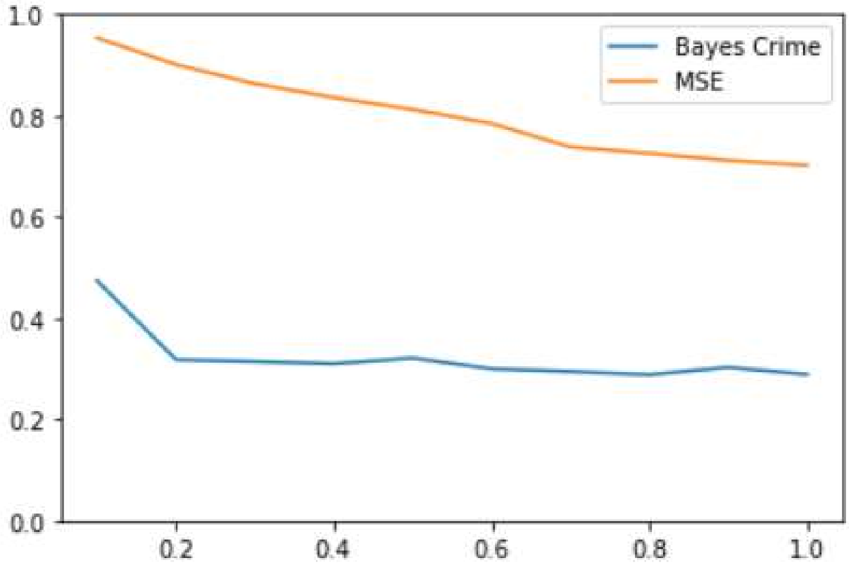
Values for crime dataset

```
{'partition': 0.1, 'alpha': (17.115748346919897-4.8112392413123525e-
30j), 'beta': (55.43127666781543+4.35640031600882e-29j), 'regulariza
tion': (0.308774204308696-3.2946525412022132e-31j)}
{'partition': 0.2, 'alpha': 16.334287029416586, 'beta': 3.4586561427
608227, 'regularization': 4.722726502779191}
{'partition': 0.3, 'alpha': 17.047085057337306, 'beta': 4.1144505963
678375, 'regularization': 4.1432226874679605}
{'partition': 0.4, 'alpha': 18.406145860229955, 'beta': 4.9746042787
79374, 'regularization': 3.700022118090225}
{'partition': 0.5, 'alpha': 17.80860565102691, 'beta': 4.41152627369
8912, 'regularization': 4.0368354501706305}
{'partition': 0.6, 'alpha': 19.53644116765738, 'beta': 4.69127660510
5255, 'regularization': 4.1644189443864725}
{'partition': 0.7, 'alpha': 18.642552046513647, 'beta': 4.3935001898
72348, 'regularization': 4.243211844963025}
{'partition': 0.8, 'alpha': 19.27321350878941, 'beta': 4.54984696577
2047, 'regularization': 4.23601357447393}
{'partition': 0.9, 'alpha': 16.95077861294148, 'beta': 4.11960383723
5635, 'regularization': 4.114662303139301}
{'partition': 1.0, 'alpha': 20.412460531028803, 'beta': 4.0412557518
33772, 'regularization': 5.051019233753366}
```

Values for housing dataset

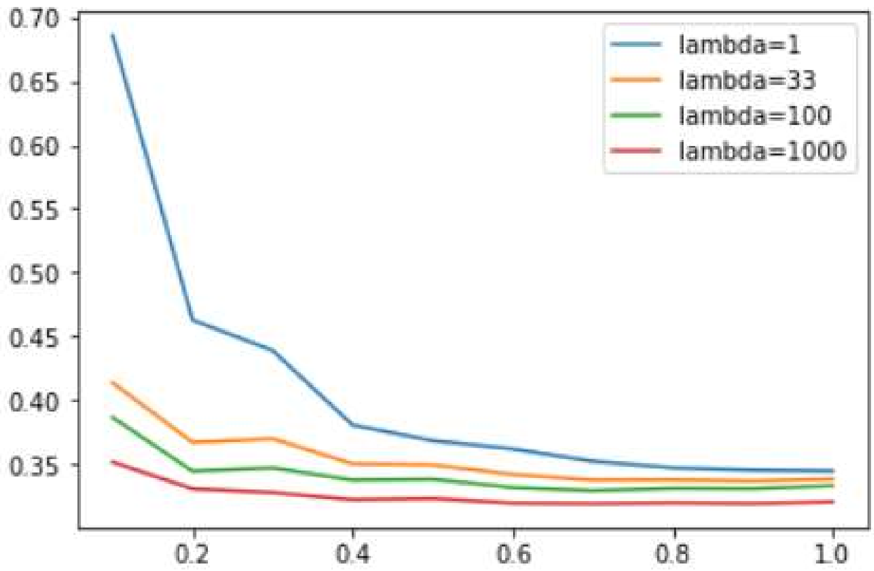


Graph for Crime dataset

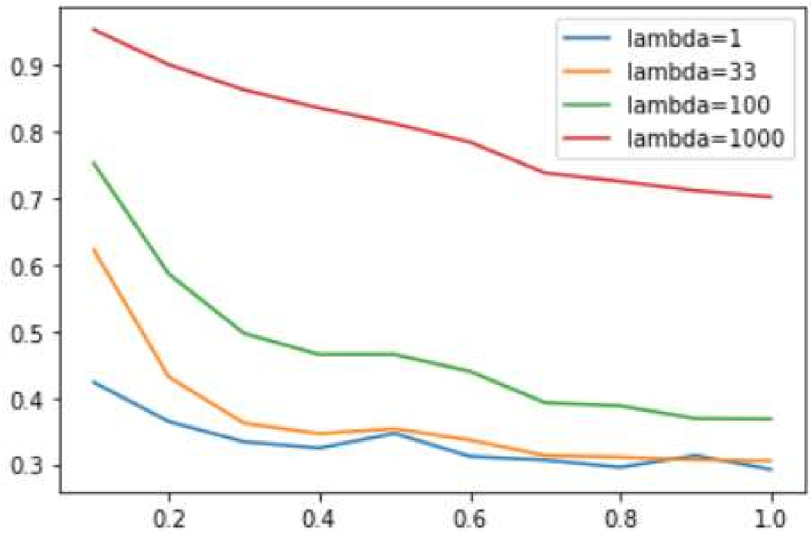


From the graph we can conclude that when size is less MSE is very high and decreases as the dataset size increases. Eventually both bayesian MSE and normal MSE converge together after a specific data size threshold

Graph for Housing dataset



Graph for crime dataset for different lamda values



Graph for housing dataset for different lamda values

We cant use universal value of lamda for different datasets because different datasets can be affected differently by the change in datasets.

Task 2:

```
logev_3
```

```
[(1, -3031.2969121330398),
 (2, -3028.564275831095),
 (3, -2695.224252770911),
 (4, -2701.4329184615654),
 (5, -2706.667939500311),
 (6, -2707.3946126834317),
 (7, -2721.462621288671),
 (8, -2732.579184186423),
 (9, -2746.881958887436),
 (10, (-2761.319226132034+0j))]
```

```
MSE_3
```

```
[(1, 39389142.58553826),
 (2, 39495762.45900575),
 (3, 148429.38836568058),
 (4, 179627.46131843395),
 (5, 186263.6807275151),
 (6, 211370.60715117436),
 (7, 184992.29642495257),
 (8, 196641.43358936673),
 (9, 546257.7759903334),
 (10, 579316.5958650279)]
```

Log evidence and MSE for f3

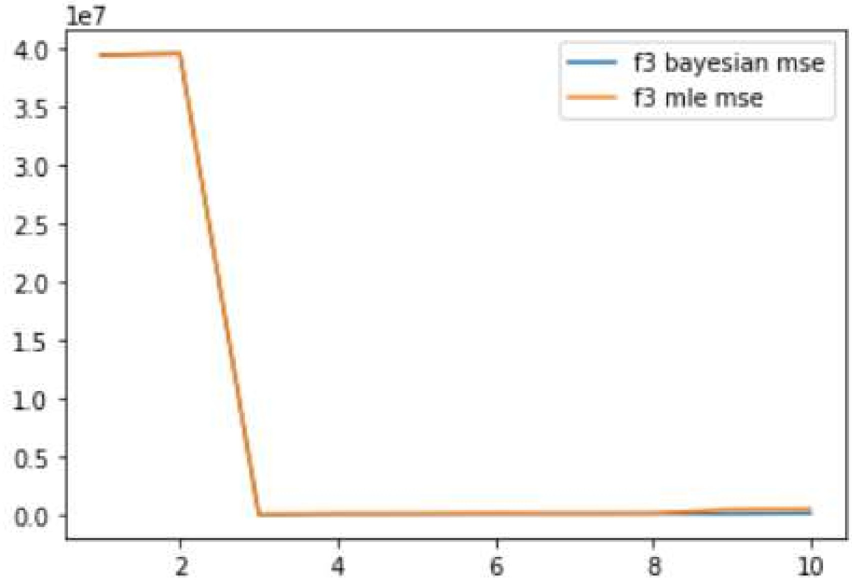
```
logev_5
```

```
[(1, -4360.422864141759),
 (2, -3912.9560540194047),
 (3, -3908.1614614004034),
 (4, -2714.974896657237),
 (5, -2721.7392525595783),
 (6, -2729.895548191721),
 (7, -2741.309541933108),
 (8, -2754.408590279996),
 (9, (-2769.3179042194065+0j)),
 (10, -2788.119185549046)]
```

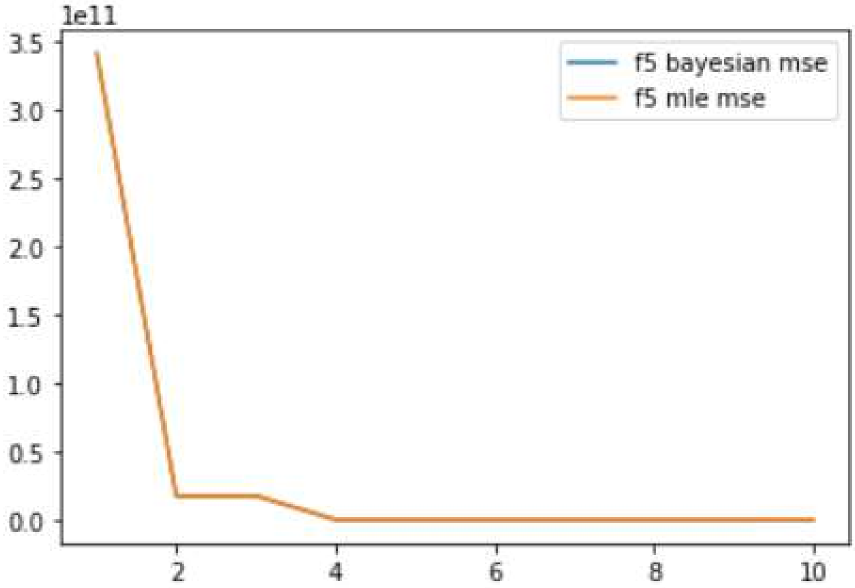
```
MSE_5
```

```
[(1, 341195638304.2565),
 (2, 17465602121.94851),
 (3, 17435655143.626442),
 (4, 61375.3499006928),
 (5, 79043.03733122443),
 (6, 92512.84710342463),
 (7, 90189.90562969688),
 (8, 126835.276487774),
 (9, 7703653.185180388),
 (10, 238757655.70912683)]
```

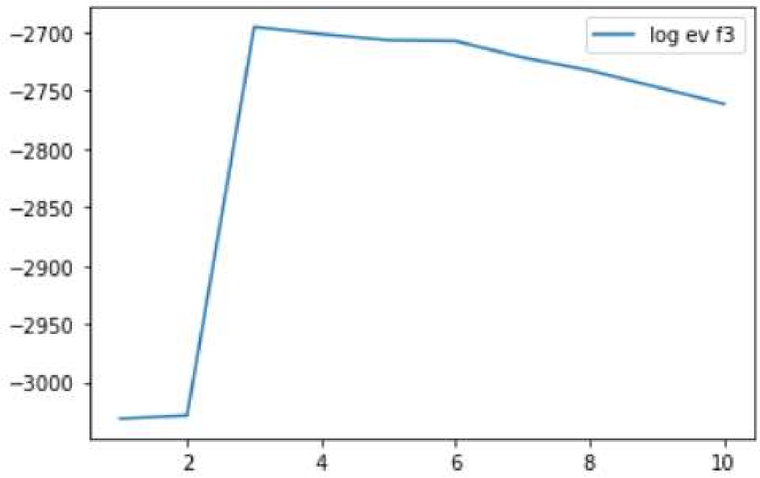
Log evidence and MSE for f5



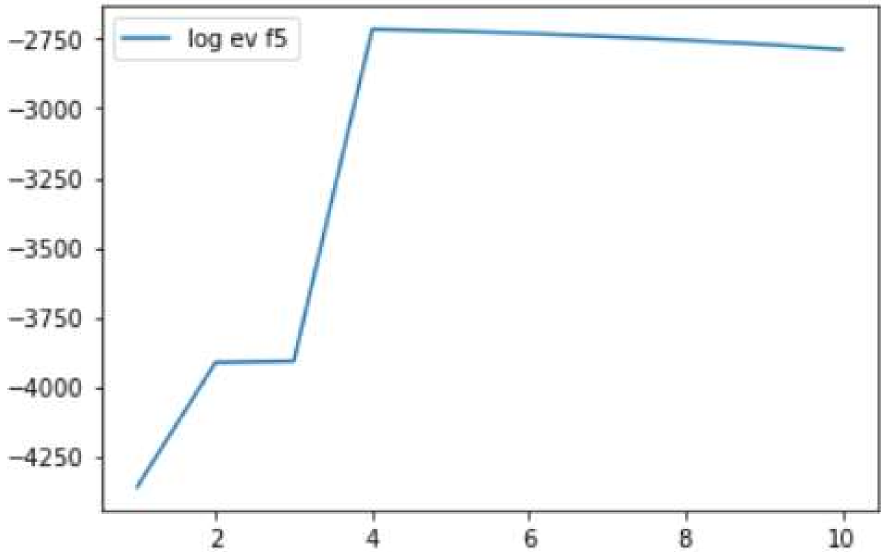
Graph for f3 bayesian and mle mse



Graph for f5 bayesian and mle mse



Graph for Log evidence of f3



Graph for log evidence of f5

The graph for regularized and non regularized are almost similar hence we can conclude that both predict similar values in this case

The evidence calculation is successful in selecting alpha, beta and the regularization

In [1]:

```
import pandas as pd
import numpy as np
from numpy import loadtxt
import matplotlib.pyplot as plt
import math
```

Task 1

Reading datasets

In [2]:

```
train_crime=open("../pp2data/train-crime.csv","rb").read().split()
train_house=open("../pp2data/train-housing.csv","rb").read().split()
trainR_crime=open("../pp2data/trainR-crime.csv","rb").read().split()
trainR_house=open("../pp2data/trainR-housing.csv","rb").read().split()
test_crime=open("../pp2data/test-crime.csv","rb").read().split()
test_house=open("../pp2data/test-housing.csv","rb").read().split()
testR_crime=open("../pp2data/testR-crime.csv","rb").read().split()
testR_house=open("../pp2data/testR-housing.csv","rb").read().split()
```

In [3]:

```
train_c=loadtxt(train_crime,delimiter=",")
trainR_c=loadtxt(trainR_crime,delimiter=",")
test_c=loadtxt(test_crime,delimiter=",")
testR_c=loadtxt(testR_crime,delimiter=",")
train_h=loadtxt(train_house,delimiter=",")
trainR_h=loadtxt(trainR_house,delimiter=",")
test_h=loadtxt(test_house,delimiter=",")
testR_h=loadtxt(testR_house,delimiter=",")
```

Calculating MSE

In [4]:

```
def calc_MSE(x,y):
    MSE=np.square(np.subtract(x,y)).mean()
    return MSE
```

Calculating MLE

In [5]:

```
def calc_MLE(x,y,z,l=0):
    pred=z.dot((np.linalg.pinv(l*np.identity(x.shape[1])+np.transpose(x).dot(x))).dot(np.tr
    return pred
```

Bayesian Function

In [6]:

```
def bayes(a,b,x,y,z):
    pred=z.dot(b*np.dot(np.dot(np.linalg.inv(a*np.identity(x.shape[1])+b*np.dot(x.T,x)),x.T
    return pred
```

In [7]:

```
pred_c=calc_MLE(train_c,trainR_c,test_c)
pred_h=calc_MLE(train_h,trainR_h,test_h)
```

In [8]:

```
MSE_c=calc_MSE(testR_c,pred_c)
MSE_c
```

Out[8]:

```
0.3450569330423569
```

In [9]:

```
MSE_h=calc_MSE(testR_h,pred_h)
MSE_h
```

Out[9]:

```
0.29443262001271514
```

Splitting Datasets according to Question

In [10]:

```
new_train_c=[]
l=len(train_c)
for i in range(0,10,1):
    new_train_c.append(train_c[0:int(l*((i+1)/10)),:])
new_trainR_c=[]
l=len(trainR_c)
for i in range(0,10,1):
    new_trainR_c.append(trainR_c[0:int(l*((i+1)/10))])

new_train_h=[]
l=len(train_h)
for i in range(0,10,1):
    new_train_h.append(train_h[0:int(l*((i+1)/10)),:])
new_trainR_h=[]
l=len(trainR_h)
for i in range(0,10,1):
    new_trainR_h.append(trainR_h[0:int(l*((i+1)/10))])
```

Model Selection

In [11]:

```
def ModelSelection(phi,t):
    N=phi.shape[0]
    d=phi.shape[1]
    a0=2
    b0=10
    a_change=1
    b_change=1
    while not (a_change<=0.001 and b_change<=0.001):
        lam=np.linalg.eigvals(b0*np.dot(phi.T,phi))
        gam=0
        for i in range(0,len(lam)):
            gam+=(lam[i]/(a0+lam[i]))
        m=b0*np.dot(np.dot(np.linalg.inv(a0*np.identity(d)+b0*np.dot(phi.T,phi)),phi.T),t)
        a1=gam/np.dot(m,m)
        b1=1/((1/(N-gam))*(np.dot(np.dot(phi,m)-t,np.dot(phi,m)-t)))
        a_change=abs(a1-a0)
        b_change=abs(b1-b0)
        a0=a1
        b0=b1
    return a0,b0,(a0/b0)
```

Crime dataset model

In [12]:

```
model_crime=[]
for i in range(10):
    d={}
    alpha,beta,regularization=ModelSelection(new_train_c[i],new_trainR_c[i])
    d["partition"]=(i+1)/10
    d["alpha"]=alpha
    d["beta"]=beta
    d["regularization"]=regularization
    model_crime.append(d)
    print(model_crime[i])
```

```
{'partition': 0.1, 'alpha': 188.13418564070818, 'beta': 3.0590610997375354,
'regularization': 61.50063026098103}
{'partition': 0.2, 'alpha': 284.7219942674846, 'beta': 2.964418873946527, 'r
egularization': 96.04647871116627}
{'partition': 0.3, 'alpha': 266.99009988195087, 'beta': 2.847152772909647,
'regularization': 93.77441998277473}
{'partition': 0.4, 'alpha': 280.4488295991778, 'beta': 2.8496845641386455,
'regularization': 98.41399049159224}
{'partition': 0.5, 'alpha': 284.08760024259936, 'beta': 2.9156222181703813,
'regularization': 97.4363545702676}
{'partition': 0.6, 'alpha': 263.4754615387708, 'beta': 2.962261804049953, 'r
egularization': 88.9440160820869}
{'partition': 0.7, 'alpha': 254.16163967645417, 'beta': 3.0875611127863833,
'regularization': 82.31793003996117}
{'partition': 0.8, 'alpha': 254.20674738385384, 'beta': 3.1241961399624962,
'regularization': 81.36708964338757}
{'partition': 0.9, 'alpha': 247.4375126121574, 'beta': 3.046255677039333, 'r
egularization': 81.22677110696198}
{'partition': 1.0, 'alpha': 239.71632213512905, 'beta': 3.0852982512664147,
'regularization': 77.69632061883588}
```

Housing dataset model

In [13]:

```
model_housing=[]
for i in range(10):
    d={}
    alpha,beta,regularization=ModelSelection(new_train_h[i],new_trainR_h[i])
    d["partition"]=(i+1)/10
    d["alpha"]=alpha
    d["beta"]=beta
    d["regularization"]=regularization
    model_housing.append(d)
    print(model_housing[i])
```

```
{'partition': 0.1, 'alpha': (17.115748346919897-4.8112392413123525e-30j), 'b
eta': (55.43127666781543+4.35640031600882e-29j), 'regularization': (0.308774
204308696-3.2946525412022132e-31j)}
{'partition': 0.2, 'alpha': 16.334287029416586, 'beta': 3.4586561427608227,
'regularization': 4.722726502779191}
{'partition': 0.3, 'alpha': 17.047085057337306, 'beta': 4.1144505963678375,
'regularization': 4.1432226874679605}
{'partition': 0.4, 'alpha': 18.406145860229955, 'beta': 4.974604278779374,
'regularization': 3.700022118090225}
{'partition': 0.5, 'alpha': 17.80860565102691, 'beta': 4.411526273698912, 'r
egularization': 4.0368354501706305}
{'partition': 0.6, 'alpha': 19.53644116765738, 'beta': 4.691276605105255, 'r
egularization': 4.1644189443864725}
{'partition': 0.7, 'alpha': 18.642552046513647, 'beta': 4.393500189872348,
'regularization': 4.243211844963025}
{'partition': 0.8, 'alpha': 19.27321350878941, 'beta': 4.549846965772047, 'r
egularization': 4.23601357447393}
{'partition': 0.9, 'alpha': 16.95077861294148, 'beta': 4.119603837235635, 'r
egularization': 4.114662303139301}
{'partition': 1.0, 'alpha': 20.412460531028803, 'beta': 4.041255751833772,
'regularization': 5.051019233753366}
```

MLE and Bayesian for crime dataset

In [14]:

```
MSE_crime=[]
l=len(train_c)
for i in range(0,10,1):
    x=train_c[0:int(round(l*((i+1)/10),0))]
    y=trainR_c[0:int(round(l*((i+1)/10),0))]
    pred=calc_MLE(x,y,test_c)
    MSE=calc_MSE(testR_c,pred)
    MSE_crime.append(((i+1)/10,MSE))
```

In [15]:

```
MSE_crime
```

Out[15]:

```
[(0.1, 1.3002479408636172),
 (0.2, 0.4967427267380541),
 (0.3, 0.45634855646314215),
 (0.4, 0.38714667059641883),
 (0.5, 0.3713426121845869),
 (0.6, 0.3644705541248856),
 (0.7, 0.35484728890387435),
 (0.8, 0.34830539217648143),
 (0.9, 0.3463928082504861),
 (1.0, 0.3450569330423569)]
```

In [16]:

```
bayes_crime=[]
for i in range(10):
    x=train_c[0:int(round(l*((i+1)/10),0))]
    y=trainR_c[0:int(round(l*((i+1)/10),0))]
    pred=bayes(model_crime[i]["alpha"],model_crime[i]["beta"],x,y,test_c)
    MSE=calc_MSE(testR_c,pred)
    bayes_crime.append(((i+1)/10,MSE))
```

In [17]:

```
bayes_crime
```

Out[17]:

```
[(0.1, 0.396984001877673 74),
 (0.2, 0.3452471828496966),
 (0.3, 0.34826092450288),
 (0.4, 0.33788161858136473),
 (0.5, 0.33862318460111307),
 (0.6, 0.3328208435831047),
 (0.7, 0.33077656253536186),
 (0.8, 0.33244628962304645),
 (0.9, 0.33214869354426235),
 (1.0, 0.3345435937647672)]
```

MLE and Bayesian for housing dataset

In [18]:

```
MSE_housing=[]
l=len(train_h)
for i in range(0,10,1):
    x=train_h[0:int(round(l*((i+1)/10),0))]
    y=trainR_h[0:int(round(l*((i+1)/10),0))]
    pred=calc_MLE(x,y,test_h)
    MSE=calc_MSE(testR_h,pred)
    MSE_housing.append(((i+1)/10,MSE))
```

In [19]:

```
MSE_housing
```

Out[19]:

```
[(0.1, 0.8678327090953848),
 (0.2, 0.6531445531536013),
 (0.3, 0.3736963444198246),
 (0.4, 0.3414166696345161),
 (0.5, 0.3644021055620389),
 (0.6, 0.32080556108532404),
 (0.7, 0.3141824758819998),
 (0.8, 0.30051685692609026),
 (0.9, 0.31907890073457246),
 (1.0, 0.29443262001271514)]
```

In [20]:

```
bayes_housing=[]
for i in range(10):
    x=train_h[0:int(round(l*((i+1)/10),0))]
    y=trainR_h[0:int(round(l*((i+1)/10),0))]
    pred=bayes(model_housing[i]["alpha"],model_housing[i]["beta"],x,y,test_h)
    MSE=calc_MSE(testR_h,pred)
    bayes_housing.append(((i+1)/10,MSE))
```

In [21]:

```
bayes_housing
```

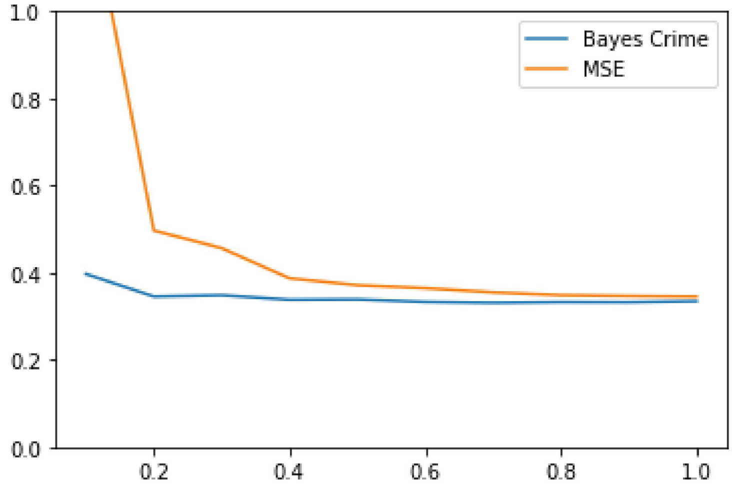
Out[21]:

```
[(0.1, (0.47393428813876126+6.53776744194863e-32j)),
 (0.2, 0.3178536604019471),
 (0.3, 0.3141356507494856),
 (0.4, 0.3098564718648854),
 (0.5, 0.32089215353385886),
 (0.6, 0.29966191365297895),
 (0.7, 0.2942475355630813),
 (0.8, 0.2879615393914638),
 (0.9, 0.3025004284077777),
 (1.0, 0.2884936266054806)]
```

Graph for Crime dataset

In [22]:

```
plt.plot(*zip(*bayes_crime),label="Bayes Crime")
plt.plot(*zip(*MSE_crime),label="MSE")
plt.ylim([0,1])
plt.legend()
plt.show()
```

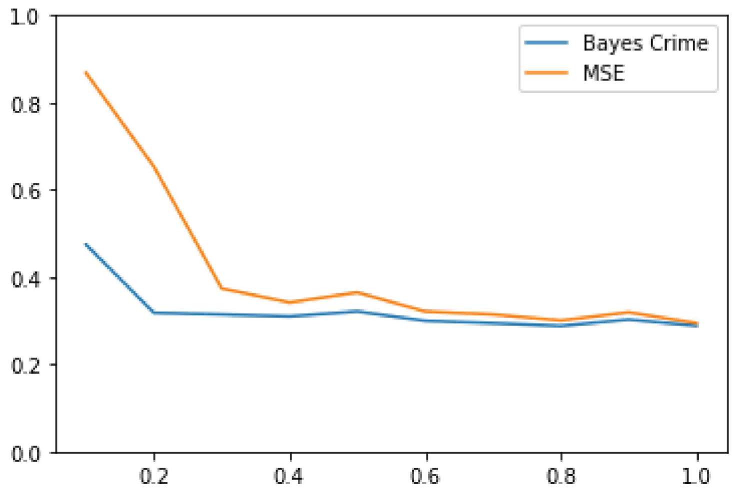


Graph for Housing Dataset

In [23]:

```
plt.plot(*zip(*bayes_housing),label="Bayes Crime")
plt.plot(*zip(*MSE_housing),label="MSE")
plt.ylim([0,1])
plt.legend()
plt.show()
```

```
C:\Users\aarya\anaconda3\lib\site-packages\matplotlib\cbook\__init__.py:129
8: ComplexWarning: Casting complex values to real discards the imaginary par
t
  return np.asarray(x, float)
```



Calculating MSE for different lamda values

In [24]:

```
lam=[1.0,33.0,100.0,1000.0]
mse_c=[]
for la in lam:
    MSE_crime=[]
    l=len(train_c)
    for i in range(0,10,1):
        x=train_c[0:int(round(l*((i+1)/10),0))]
        y=trainR_c[0:int(round(l*((i+1)/10),0))]
        pred=calc_MLE(x,y,test_c,la)
        MSE=calc_MSE(testR_c,pred)
        MSE_crime.append(((i+1)/10,MSE))
    mse_c.append(MSE_crime)
```

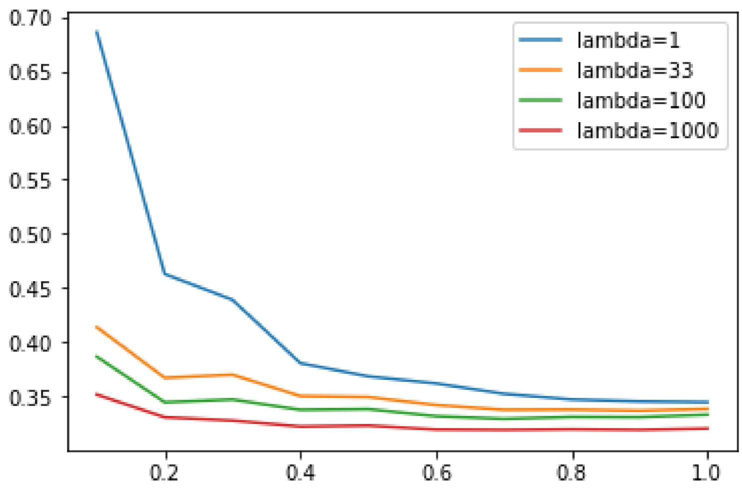
In [25]:

```
lam=[1.0,33.0,100.0,1000.0]
mse_h=[]
for la in lam:
    MSE_housing=[]
    l=len(train_h)
    for i in range(0,10,1):
        x=train_h[0:int(round(l*((i+1)/10),0))]
        y=trainR_h[0:int(round(l*((i+1)/10),0))]
        pred=calc_MLE(x,y,test_h,la)
        MSE=calc_MSE(testR_h,pred)
        MSE_housing.append(((i+1)/10,MSE))
    mse_h.append(MSE_housing)
```

Graph for Crime datset

In [26]:

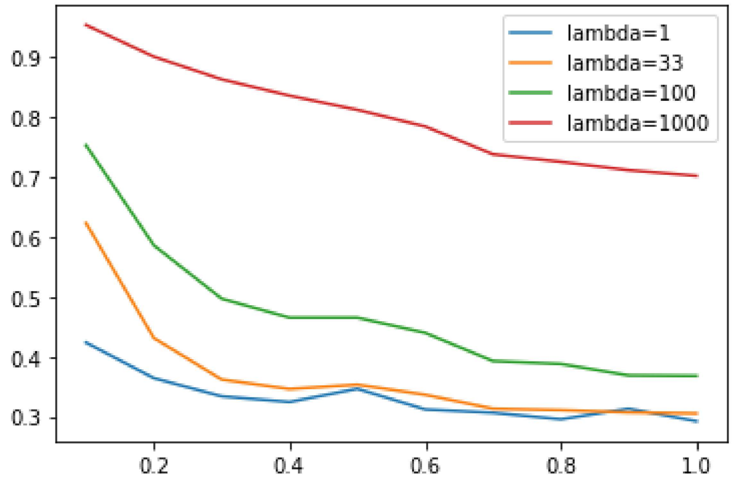
```
plt.plot(*zip(*mse_c[0]),label="lambda=1")
plt.plot(*zip(*mse_c[1]),label="lambda=33")
plt.plot(*zip(*mse_c[2]),label="lambda=100")
plt.plot(*zip(*mse_c[3]),label="lambda=1000")
plt.legend()
plt.show()
```



Graph for Housing dataset

In [27]:

```
plt.plot(*zip(*mse_h[0]),label="lambda=1")
plt.plot(*zip(*mse_h[1]),label="lambda=33")
plt.plot(*zip(*mse_h[2]),label="lambda=100")
plt.plot(*zip(*mse_h[3]),label="lambda=1000")
plt.legend()
plt.show()
```



Task 2

Creating datsets

In [28]:

```
train_f3=open("../pp2data/train-f3.csv","rb").read().split()
train_f5=open("../pp2data/train-f5.csv","rb").read().split()
trainR_f3=open("../pp2data/trainR-f3.csv","rb").read().split()
trainR_f5=open("../pp2data/trainR-f5.csv","rb").read().split()
test_f3=open("../pp2data/test-f3.csv","rb").read().split()
test_f5=open("../pp2data/test-f5.csv","rb").read().split()
testR_f3=open("../pp2data/testR-f3.csv","rb").read().split()
testR_f5=open("../pp2data/testR-f5.csv","rb").read().split()
```

In [29]:

```
train_3=loadtxt(train_f3,delimiter=",")
trainR_3=loadtxt(trainR_f3,delimiter=",")
test_3=loadtxt(test_f3,delimiter=",")
testR_3=loadtxt(testR_f3,delimiter=",")
train_5=loadtxt(train_f5,delimiter=",")
trainR_5=loadtxt(trainR_f5,delimiter=",")
test_5=loadtxt(test_f5,delimiter=",")
testR_5=loadtxt(testR_f5,delimiter=",")
```

In [30]:

```
def bayes2(a,b,x,y,z):
    pred=z.dot(b*np.dot(np.dot(np.linalg.inv(a*np.identity(x.shape[1])+b*np.dot(x.T,x)),x.T
    mn=(b*np.dot(np.dot(np.linalg.inv(a*np.identity(x.shape[1])+b*np.dot(x.T,x)),x.T),y))
    sn=(np.linalg.inv(a*np.identity(x.shape[1])+b*np.dot(x.T,x)),x.T)
    return pred,mn,sn
```

Calculating log Evidence and MSE for f3

In [31]:

```
logev_3=[]
MSE_3=[]
bayes_3=[]
for i in range(10):

    new_train_3=np.hstack((np.ones((train_3.shape[0],1)),train_3.reshape(train_3.shape[0],1
    new_test_3=np.hstack((np.ones((test_3.shape[0],1)),test_3.reshape(test_3.shape[0],1)))
    for j in range(2,i+2):
        new_train_3=np.hstack((new_train_3,(new_train_3[:,1]**(j)).reshape(new_train_3.shap
        new_test_3=np.hstack((new_test_3,(new_test_3[:,1]**(j)).reshape(new_test_3.shape[0]

    a,b,l=ModelSelection(new_train_3,trainR_3)

    bayes,mn,sn=bayes2(a,b,new_train_3,trainR_3,new_test_3)

    e=np.dot((trainR_3-np.dot(new_train_3,mn)).T,(trainR_3-np.dot(new_train_3,mn)))*b/2
    e1=(a*np.dot(mn.T,mn))/2
    e2=np.log(np.linalg.det(np.linalg.inv(sn[0])))
    log=(((new_train_3.shape[1]/2*np.log(alpha))+((300/2)*np.log(b))-e-e1-(e2/2)-(300*np.lo
    logev_3.append((i+1,log))


    MSE=calc_MSE(testR_3,bayes)
    bayes_3.append((i+1,MSE))

    pred=calc_MLE(new_train_3,trainR_3,new_test_3)
    MSE=calc_MSE(testR_3,pred)
    MSE_3.append((i+1,MSE))
```

In [32]:

```
logev_3
```

Out[32]:

```
[(1, -3031.2969121330398),
 (2, -3028.564275831095),
 (3, -2695.224252770911),
 (4, -2701.4329184615654),
 (5, -2706.667939500311),
 (6, -2707.3946126834317),
 (7, -2721.462621288671),
 (8, -2732.579184186423),
 (9, -2746.881958887436),
 (10, (-2761.319226132034+0j))]
```

In [33]:

```
MSE_3
```

Out[33]:

```
[(1, 39389142.58553826),
 (2, 39495762.45900575),
 (3, 148429.38836568058),
 (4, 179627.46131843395),
 (5, 186263.6807275151),
 (6, 211370.60715117436),
 (7, 184992.29642495257),
 (8, 196641.43358936673),
 (9, 546257.7759903334),
 (10, 579316.5958650279)]
```

Calculating Log evidence and MSE for f5

In [34]:

```
logev_5=[]
MSE_5=[]
bayes_5=[]
for i in range(10):

    new_train_5=np.hstack((np.ones((train_5.shape[0],1)),train_5.reshape(train_5.shape[0],1
    new_test_5=np.hstack((np.ones((test_5.shape[0],1)),test_5.reshape(test_5.shape[0],1)))
    j=2
    while j <=i+2:
        new_train_5=np.hstack((new_train_5,(new_train_5[:,1]**(j)).reshape(new_train_5.shap
        new_test_5=np.hstack((new_test_5,(new_test_5[:,1]**(j)).reshape(new_test_5.shape[0]
        j+=1
    a,b,l=ModelSelection(new_train_5,trainR_5)

    bayes,mn,sn=bayes2(a,b,new_train_5,trainR_5,new_test_5)

    e=np.dot((trainR_5-np.dot(new_train_5,mn)).T,(trainR_5-np.dot(new_train_5,mn)))*b/2
    e1=(a*np.dot(mn.T,mn))/2
    e2=np.log(np.linalg.det(np.linalg.inv(sn[0])))
    log=(((new_train_5.shape[1]/2*np.log(alpha))+((300/2)*np.log(b))-e-e1-(e2/2)-(300*np.lo
    logev_5.append((i+1,log))


    MSE=calc_MSE(testR_5,bayes)
    bayes_5.append((i+1,MSE))

    pred=calc_MLE(new_train_5,trainR_5,new_test_5)
    MSE=calc_MSE(testR_5,pred)
    MSE_5.append((i+1,MSE))
```

In [35]:

```
logev_5
```

Out[35]:

```
[(1, -4360.422864141759),
 (2, -3912.9560540194047),
 (3, -3908.1614614004034),
 (4, -2714.974896657237),
 (5, -2721.7392525595783),
 (6, -2729.895548191721),
 (7, -2741.309541933108),
 (8, -2754.408590279996),
 (9, (-2769.3179042194065+0j)),
 (10, -2788.119185549046)]
```

In [36]:

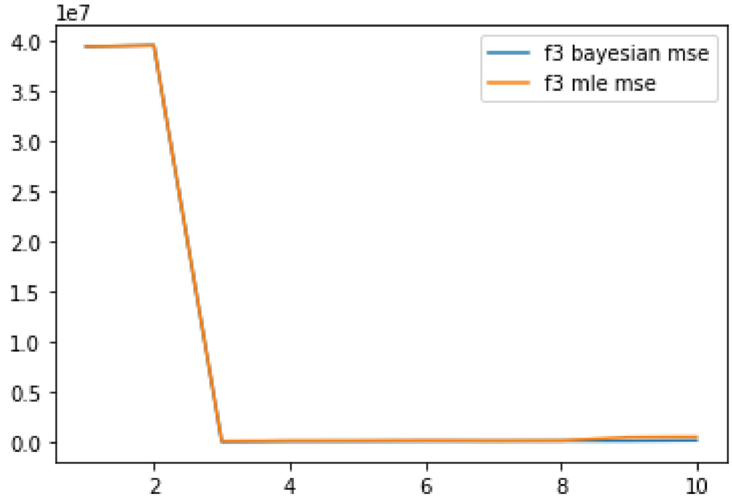
```
MSE_5
```

Out[36]:

```
[(1, 341195638304.2565),
 (2, 17465602121.94851),
 (3, 17435655143.626442),
 (4, 61375.3499006928),
 (5, 79043.03733122443),
 (6, 92512.84710342463),
 (7, 90189.90562969688),
 (8, 126835.276487774),
 (9, 7703653.185180388),
 (10, 238757655.70912683)]
```

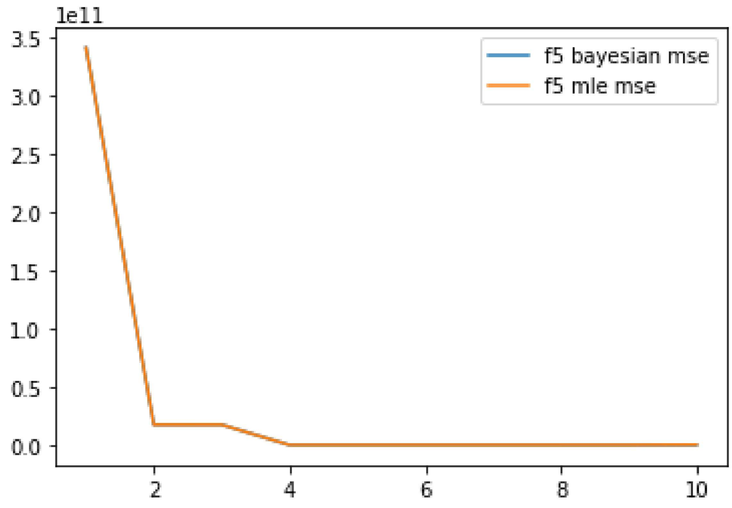
In [37]:

```
plt.plot(*zip(*bayes_3),label='f3 bayesian mse')
plt.plot(*zip(*MSE_3),label='f3 mle mse')
plt.legend()
plt.show()
```



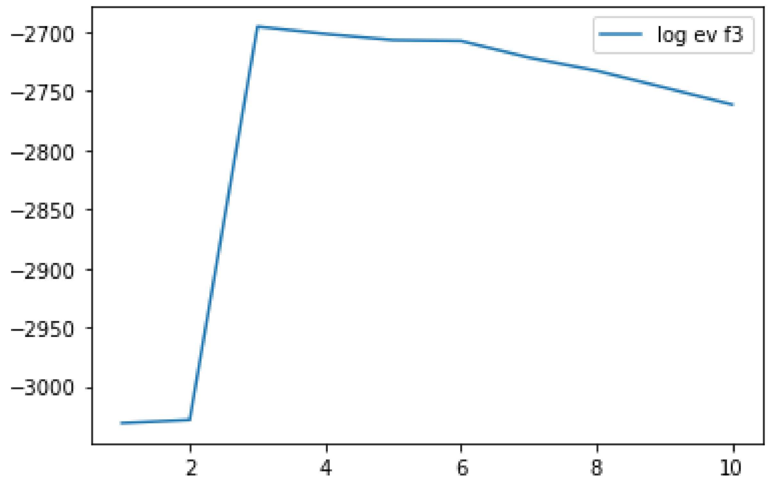
In [38]:

```
plt.plot(*zip(*bayes_5),label='f5 bayesian mse')
plt.plot(*zip(*MSE_5),label='f5 mle mse')
plt.legend()
plt.show()
```



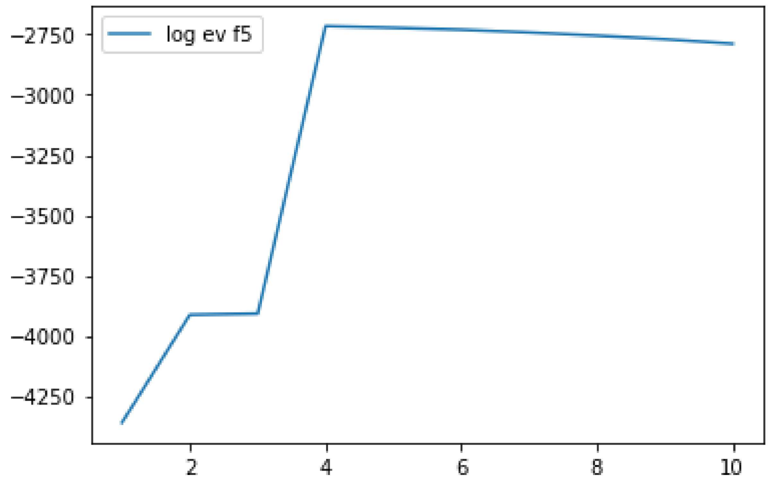
In [39]:

```
plt.plot(*zip(*logev_3),label='log ev f3')
plt.legend()
plt.show()
```



In [40]:

```
plt.plot(*zip(*logev_5),label='log ev f5')
plt.legend()
plt.show()
```



In [ ]:

The assignment is written in ipynb format and can be run using the jupyter notebook

The data files are attached to the directory, in case of data not loading please check the path

There are no specific instructions to run the file