ML Assignment 2

Importing Pyhton libraries

In [30]:
```
import numpy as np
```

In [31]:
```
import matplotlib.pyplot as plt
```

Randomly generating samples using normal distribution

In [32]:
```
from scipy.stats import norm
```

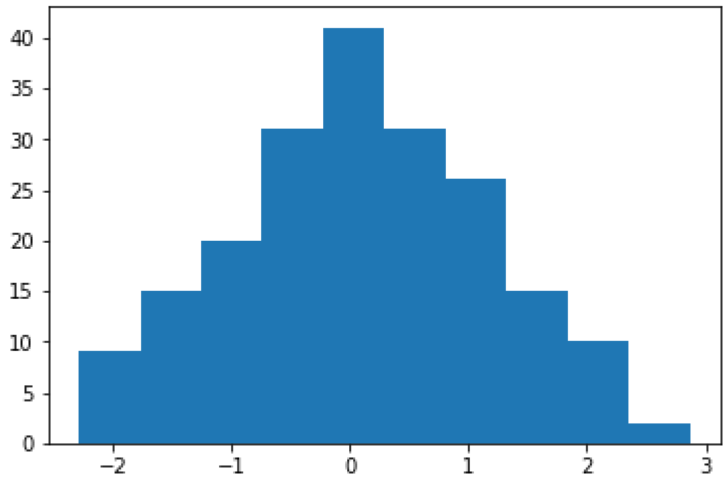
In [33]:
```
from scipy.stats import expon
```

In [34]:
```
from scipy.stats import bernoulli
```

In [35]:
```
data1=norm.rvs(0,1,200)
plt.hist(data1)
```

Out[35]:
```
(array([  9.,  15.,  20.,  31.,  41.,  31.,  26.,  15.,  10.,   2.]),
 array([-2.28096773, -1.76638143, -1.25179513, -0.73720882, -0.2226225
2,
         0.29196378,  0.80655008,  1.32113638,  1.83572268,  2.3503089
8,
         2.86489528]),
 <a list of 10 Patch objects>)
```

In [36]:
```
plt.show()
```



Fitting the data

In [37]:
```
fit1=norm.fit(data1)
```

Returns two parameters for the data in array form

In [38]:
```
z=np.linspace(-4,4,90)
```

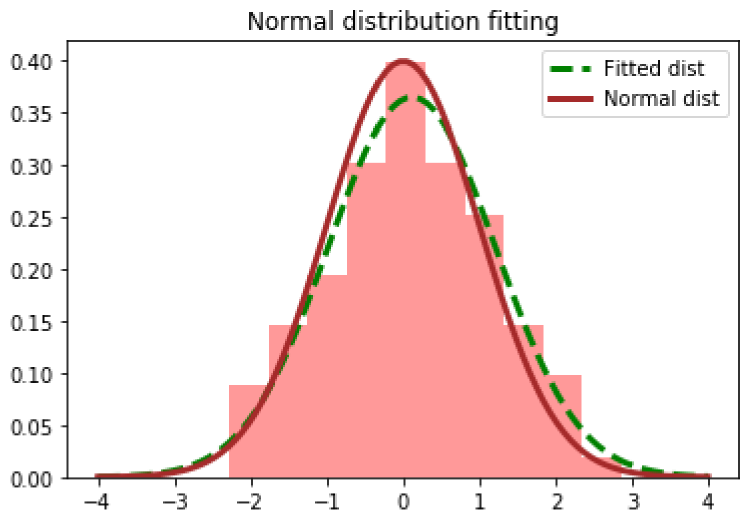
Two parameters are the mean and standard deviation of the number of samples to generate

In [39]:
```
pdf_fit=norm.pdf(z,fit1[0],fit1[1])
```

Generate probability density function (Fitted distribution)

In [40]:
```
pdf_normal=norm.pdf(z)
```

In [41]:
```
plt.plot(z,pdf_fit,"green",label="Fitted dist",linestyle="dashed", linew
idth=3)
plt.plot(z,pdf_normal,"brown",label="Normal dist", linewidth=3)
plt.hist(data1,normed=2,color="red",alpha=.4) #alpha, from 0 (transparen
t) to 1 (o
plt.title("Normal distribution fitting")
# insert a legend in the plot (using label)
plt.legend()
plt.show()
```



Summary:

From the above graph, we can observe that the fitted normal distribution almost fits the normal distribution curve overa range of values. Hence, we can conclude that as we increase the number of observations the distribution of the given data will fit more accurately.

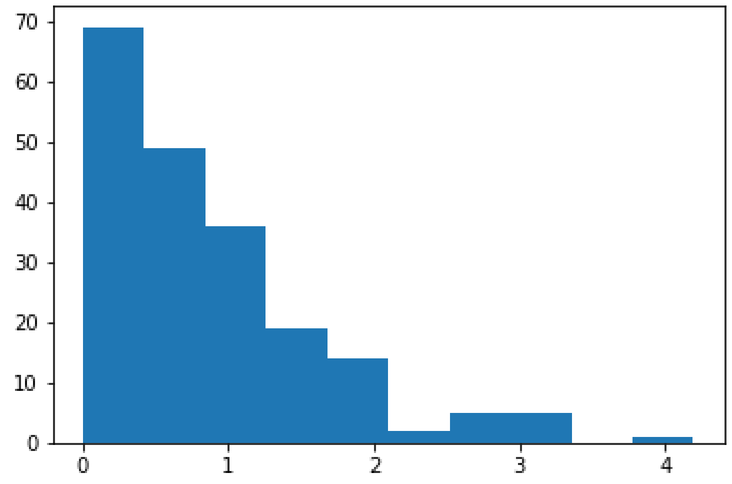
Randomly generating samples using exponential distribution

In [42]:
```
data2=expon.rvs(loc=0,scale=1,size=200)
```

```
In [43]: data2
```

```
Out[43]: array([  5.98069146e-01,   1.82895228e-01,   6.70006966e-01,
                  8.39702008e-02,   1.27675072e-01,   1.16374280e-01,
                  1.98625467e+00,   5.56402770e-01,   3.90841718e-01,
                  1.51779923e+00,   1.12382166e-01,   3.75825610e-01,
                  2.06937483e-01,   1.16096530e-01,   2.83541268e-01,
                  1.29786048e-01,   5.59111646e-01,   9.77279096e-01,
                  9.65673777e-01,   5.04728235e-01,   5.41419124e-01,
                  2.46989799e-01,   1.90538141e-01,   1.09905677e+00,
                  3.00336666e+00,   4.45285122e-01,   2.20022052e-01,
                  1.06668288e+00,   5.63795259e-01,   6.40787404e-01,
                  2.50123456e-01,   2.03743026e-01,   7.98255039e-02,
                  1.34781088e+00,   1.77605344e+00,   2.85977160e+00,
                  2.67716103e+00,   4.21965306e-01,   5.75513489e-02,
                  1.14826003e+00,   1.85447510e+00,   1.74195298e+00,
                  9.08460215e-01,   1.38310998e-01,   1.27436958e+00,
                  6.71943145e-01,   2.19546405e-01,   8.46774961e-01,
                  2.06021320e-02,   1.45332397e+00,   3.12112702e+00,
                  2.38866212e-01,   6.03784393e-02,   9.95907619e-01,
                  5.23391523e-01,   1.45748154e+00,   5.39373943e-03,
                  1.96319514e+00,   1.35376652e+00,   9.12801566e-02,
                  6.51401265e-01,   7.13791959e-01,   5.91370057e-01,
                  9.20531979e-01,   5.22517550e-01,   5.37594504e-01,
                  1.66427138e+00,   1.73925068e-01,   4.31494737e-01,
                  4.19516683e+00,   1.37870501e+00,   7.59154772e-01,
                  1.06295136e+00,   3.13776107e-01,   9.41782862e-01,
                  1.29844878e+00,   1.94813270e-01,   8.06050613e-01,
                  7.97597579e-01,   7.80242012e-01,   4.85512107e-01,
                  7.10514047e-01,   1.27775201e+00,   1.18811209e-01,
                  4.99463838e-01,   2.37314563e-01,   2.38415445e-01,
                  3.31801563e+00,   2.58249370e-01,   1.01208187e+00,
                  3.10240334e+00,   8.83269307e-01,   3.57489315e-01,
                  2.39476682e-01,   2.27400214e-01,   2.10325312e-01,
                  1.68590518e+00,   1.80301798e+00,   6.33243417e-01,
                  1.18073573e+00,   8.56730306e-01,   1.28768569e+00,
                  3.26222235e-01,   6.32448613e-01,   2.14003304e+00,
                  1.54291139e+00,   1.00331571e+00,   5.56435403e-01,
                  6.06362666e-02,   4.68260565e-01,   1.85179556e+00,
                  3.23282637e+00,   5.83557317e-01,   1.06601475e-02,
                  1.14535803e+00,   9.97102336e-02,   1.70944951e-01,
                  1.03073983e+00,   9.29978592e-01,   4.38891152e-01,
                  8.33147096e-02,   2.61657421e-01,   4.85386054e-01,
                  1.74726047e+00,   7.06398902e-01,   5.92074587e-02,
                  8.53240196e-01,   1.50228638e+00,   5.04431520e-01,
                  7.76574325e-02,   1.72927692e+00,   3.53179453e-01,
                  1.16102946e+00,   6.25746299e-01,   7.51553028e-01,
                  6.69736156e-01,   6.82063254e-02,   9.77005010e-01,
                  9.02929509e-01,   6.81859691e-01,   2.17750569e+00,
                  2.80151620e+00,   5.82717619e-01,   8.69506593e-01,
                  1.33428054e+00,   3.64834027e-02,   6.14879332e-01,
                  4.05906755e-03,   3.60663116e-01,   2.18407285e-01,
                  1.37556178e+00,   9.95335012e-01,   5.29417674e-01,
                  7.39919738e-02,   9.40524290e-02,   2.04263314e+00,
                  1.86903944e+00,   8.74042186e-03,   8.46129756e-01,
                  1.25141516e-01,   2.49528958e-01,   3.79864175e-02,
                  3.72284341e-01,   9.83360412e-01,   2.58567889e-01,
                  1.35207398e+00,   8.20690868e-01,   6.03501217e-02,
                  8.04341494e-01,   6.33705252e-01,   1.41556796e+00,
                  1.68850425e+00,   1.04683356e+00,   4.26343769e-01,
                  7.05529622e-02,   2.06744166e-01,   1.01609697e+00,
                  6.78717525e-01,   1.40513196e+00,   9.27180065e-01,
                  3.06203967e-01,   6.30102132e-01,   2.55479613e+00,
                  1.06970243e+00,   2.40287038e-01,   4.51033389e-02,
                  6.67174942e-01,   1.34341282e-01,   1.34285754e+00,
                  3.16440890e-01,   8.70434471e-01,   1.08650861e+00,
                  1.00030043e+00,   9.49696634e-01,   3.82320489e-01,
                  7.37148659e-01,   5.15477849e-01,   9.39149265e-01,
                  2.75895594e+00,   2.09410526e+00])
```

In [44]:
```
plt.hist(data2)
plt.show()
```



Returns two parameters for the data in array form

In [45]:
```
fit2=expon.fit(data2)
```

Two parameters are the mean and standard deviation of the number of samples to generate

In [46]:
```
y=np.linspace(0,5,100)
```

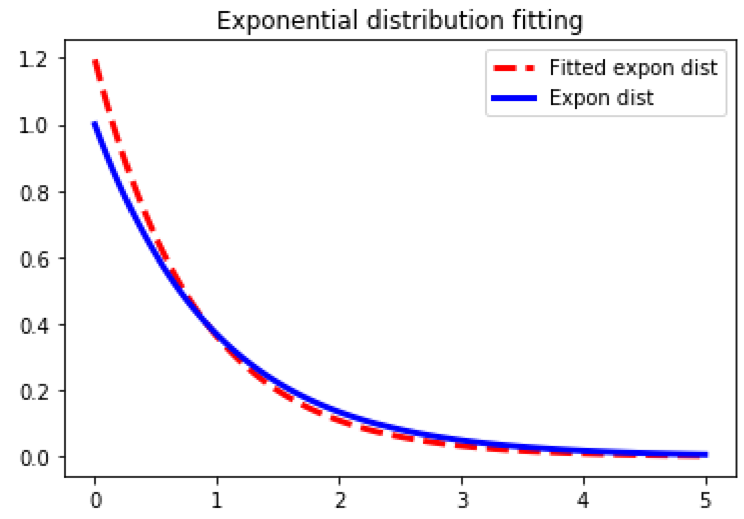
Generate probability density function (Fitted exponential distribution)

In [47]:
```
pdf_fit=expon.pdf(y,scale=fit2[1])
```

Generate probability density function (Exponential distribution)

In [48]:
```
pdf_expon=expon.pdf(y)
```

In [49]:
```
plt.plot(y,pdf_fit,"red",label="Fitted expon dist",linestyle="dashed", l
inewidth=3)
plt.plot(y,pdf_expon,"blue",label="Expon dist", linewidth=3)
#plt.hist(data2,normed=2,color="yellow",alpha=.4) #alpha, from 0 (transp
arent) to
plt.title("Exponential distribution fitting")
# insert a legend in the plot (using label)
plt.legend()
plt.show()
```

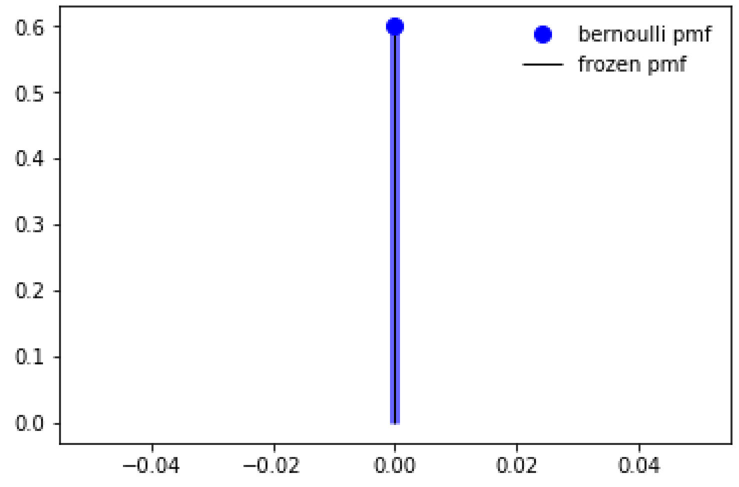


Summary:

From the above graph, we can observe that the fitted exponential distribution almost fits the normal distribution curve overall range of values.The accuracy of the fit increases as we increase the number of sampled datapoints.

Randomly generating samples using bernoulli distribution

```
In [50]: fig, ax = plt.subplots(1, 1)
         u = 0.4 # it has to be between 0,1
         b = np.arange(bernoulli.ppf(0.01, u),
         bernoulli.ppf(0.99,u))
         ax.plot(b, bernoulli.pmf(b, u), 'bo', ms=8, label='bernoulli pmf')
         ax.vlines(b, 0, bernoulli.pmf(b, u), colors='b', lw=5, alpha=0.6)
         rv = bernoulli(u)
         ax.vlines(b, 0, rv.pmf(b), colors='k', linestyles='-', lw=1,
         label='frozen pmf')
         ax.legend(loc='best', frameon=False)
         plt.show()
```



Check accuracy of cdf and ppf

```
In [51]: prbl = bernoulli.cdf(b, u)
         np.allclose(b, bernoulli.ppf(prbl, u))

Out[51]: True
```

Summary:

From the above graph, we can observe that the bernoulli distribution fits the frozen curve and the result of accuracy check of cdf and pdf shows that the fit of the data distribution for the sampled datapoints is accurate.

References

https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html (https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html)

https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.exponential.html (https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.exponential.html)

https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.bernoulli.html (https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.bernoulli.html)

https://stackoverflow.com (https://stackoverflow.com)