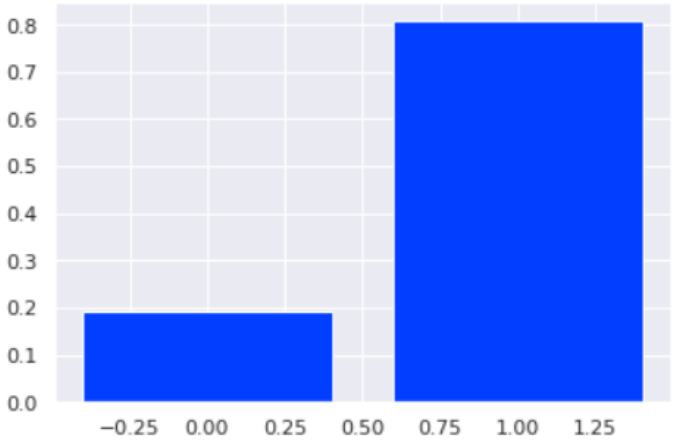
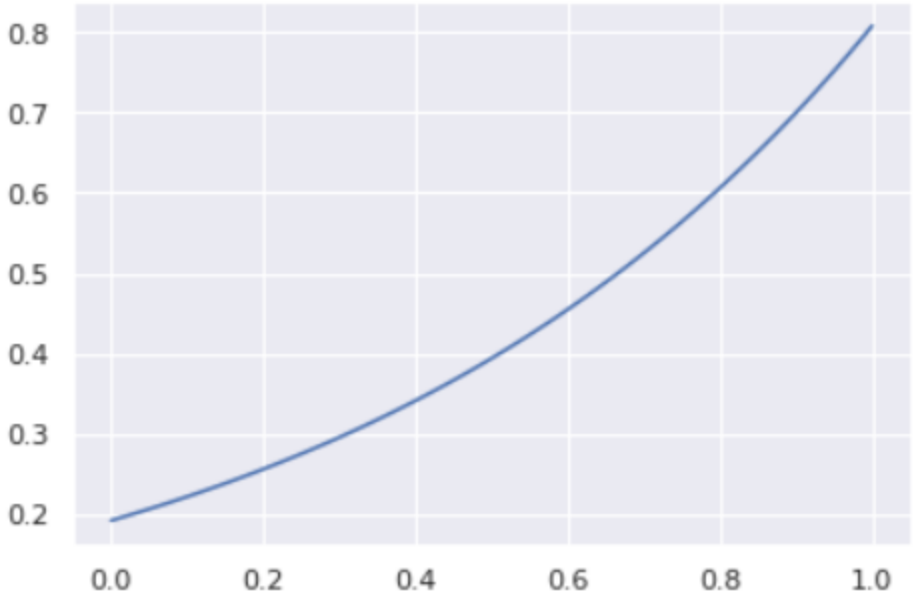
Week 3

In [3]:
```
counter = Counter(train_data)
plt.bar(counter.keys(), list(map(lambda a: a / train_size, counter.values())))
```

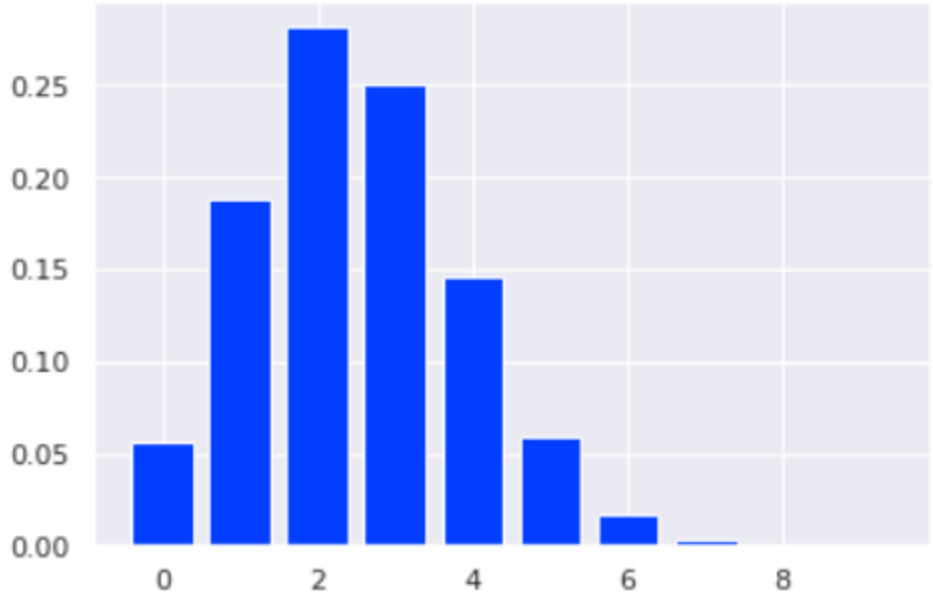
Out[3]: <BarContainer object of 2 artists>



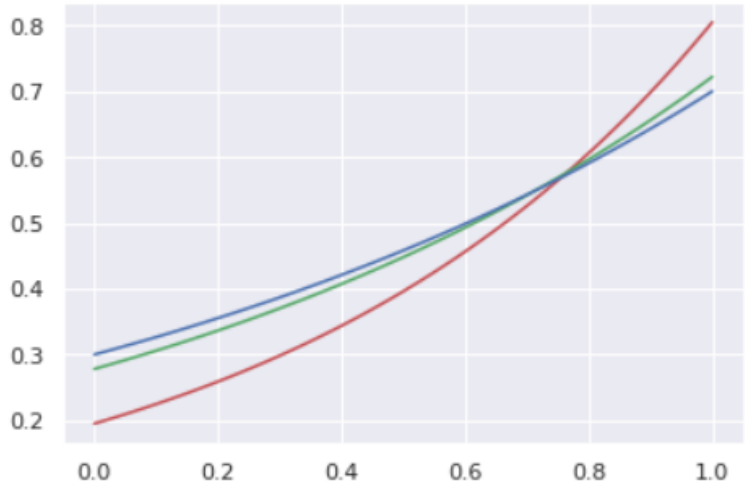
In [7]:
```
display_bernoulli(mu_ml)
```



In [10]:
```
display_binomial(10, 0.25)
```



In [12]:
```
display_bernoulli(
    mu_maximum_posterior(sum(train_data), train_size, 0.1, 0.1), 'r'
)
display_bernoulli(
    mu_maximum_posterior(sum(train_data), train_size, 70, 30), 'g'
)
display_bernoulli(
    mu_maximum_posterior(sum(train_data), train_size, 7000, 3000), '
)
```



In [13]:
```
np.random.seed(26)

def generate_multinomial_data(size
    numbers = np.random.choice(len

    result = np.zeros((numbers.siz
    result[np.arange(numbers.size)
    return result

probabilities=[0.5, 0.1, 0.2, 0.1,
multinomial_data = generate_multin
multinomial_data
```

Out[13]:
```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 1., 0., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.],
       [1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 0., 1.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 1., 0.]])
```

```
multi_mu_ml = multinomial_mu_maximum_likelihood(multinomial_data)
multi_mu_ml
```

Out[15]: array([0.19230769, 0.11538462, 0.23076923, 0.23076923, 0.23076923])

In [17]:
```
print(multinomial_mu_maximum_posterior(ms, len(multinomial_data), [5, 1, 2, 1, 1]))
print(multinomial_mu_maximum_posterior(ms, len(multinomial_data), [50, 10, 20, 10, 10]))
print(multinomial_mu_maximum_posterior(ms, len(multinomial_data), [500, 100, 200, 100, 100]))
```

```
[0.27777778 0.11111111 0.22222222 0.19444444 0.19444444]
[0.43650794 0.1031746  0.20634921 0.12698413 0.12698413]
[0.49220273 0.10038986 0.20077973 0.10331384 0.10331384]
```

In [18]:
```
print(probabilities)
```

```
[0.5, 0.1, 0.2, 0.1, 0.1]
```

4.3) Display data

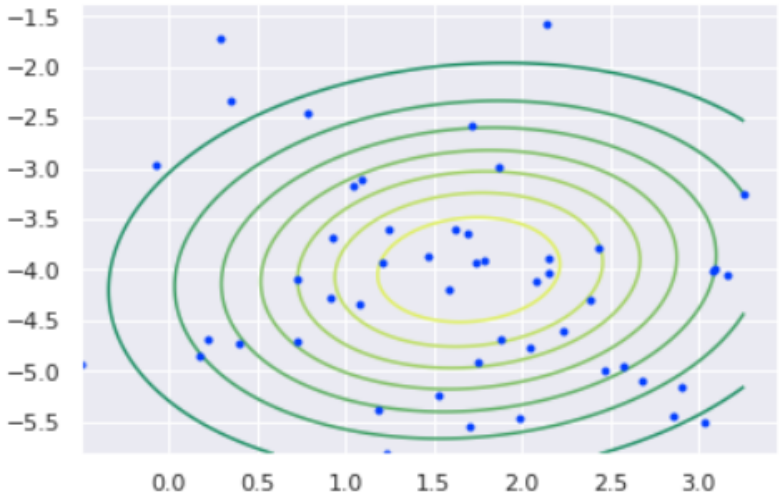
In [21]:
```
plt.plot(multi_gaussian_data[:, 0], multi_gaussian_data[:, 1], '.')

def display_gaussian_contour(data, mean, covariance, cmap='summer'):

    lx = min(data[:, 0])
    rx = max(data[:, 0])
    by = min(data[:, 1])
    uy = max(data[:, 1])

    x, y = np.mgrid[lx:rx:.01, by:uy:.01]
    pos = np.dstack((x, y))
    plt.contour(x, y, multivariate_normal(mean, covariance).pdf(pos)

display_gaussian_contour(multi_gaussian_data, target_mean, target_co
```

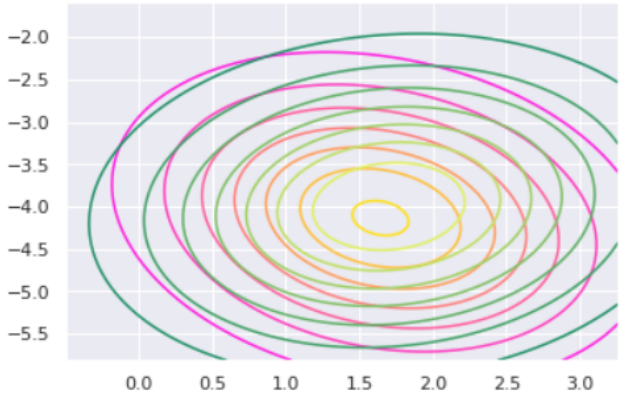


```
g_cov_ml = gaussian_covariance_maximum_likelihood(multi_gaussian_data, g_mu_ml)
g_cov_ml
```

Out[23]:
```
array([[ 0.84757357, -0.18427121],
       [-0.18427121,  0.97482937]])
```

4.4.2) Display distribution

In [24]:
```
display_gaussian_contour(multi_gaussian_data, g_mu_ml, g_cov_ml, cmap='spring')
display_gaussian_contour(multi_gaussian_data, target_mean, target_cov, cmap='summer')
```



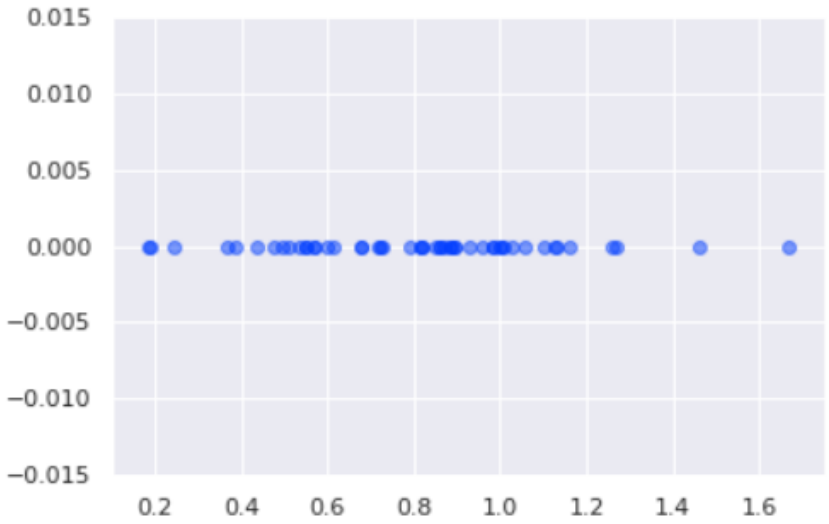
4.5.1) Generate data

In [25]:
```
def generate_gaussian_data(size, mean, variance):
    return np.array(norm(mean, sqrt(variance)).rvs(size=size, ran

g_mean = 0.8
g_variance = 0.1
gaussian_data = generate_gaussian_data(50, g_mean, g_variance)

plt.scatter(gaussian_data, [0] * len(gaussian_data), alpha=0.5)
```

Out[25]: <matplotlib.collections.PathCollection at 0x7f0942be4a10>




```
g_mu_map, g_mu_var_map = gaussian_mu_maximum_posterior(gaussian_data, 0, g_variance, g_variance)
g_mu_map, g_mu_var_map
```

Out[26]: (0.7873049121325897, 0.00196078431372549)

```
display_mu_map(gaussian_data[:0], 0, g_variance, g_variance)
display_mu_map(gaussian_data[:1], 0, g_variance, g_variance)
display_mu_map(gaussian_data[:10], 0, g_variance, g_variance)
display_mu_map(gaussian_data, 0, g_variance, g_variance)
```