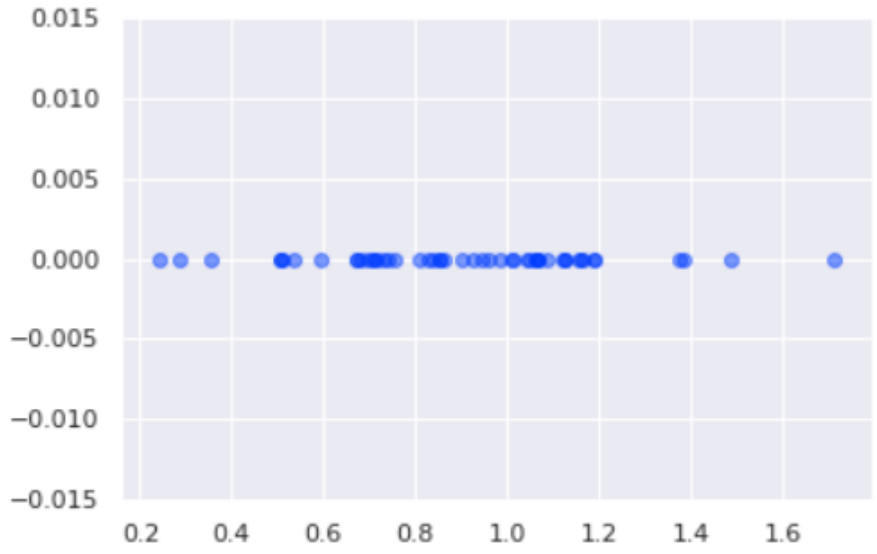
Week 4

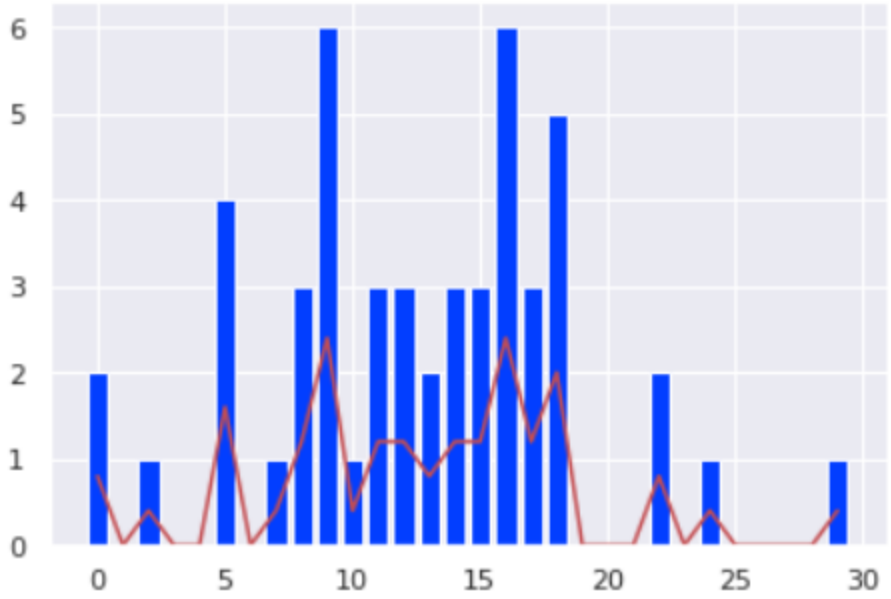
Generate data

In [29]:
```
def generate_data_1D(size, means, variances, pis):
    result = 0

    for i, (mean, variance, pi) in enumerate(zip(means, variar
        result += pi * np.array(norm(mean, sqrt(variance)).rvs

    return result

means = [0.4, 2.0]
variances = [0.2, 0.1]
pis = [0.7, 0.3]
data_1D = generate_data_1D(50, means, variances, pis)

plt.scatter(data_1D, [0] * len(data_1D), alpha=0.5)
```

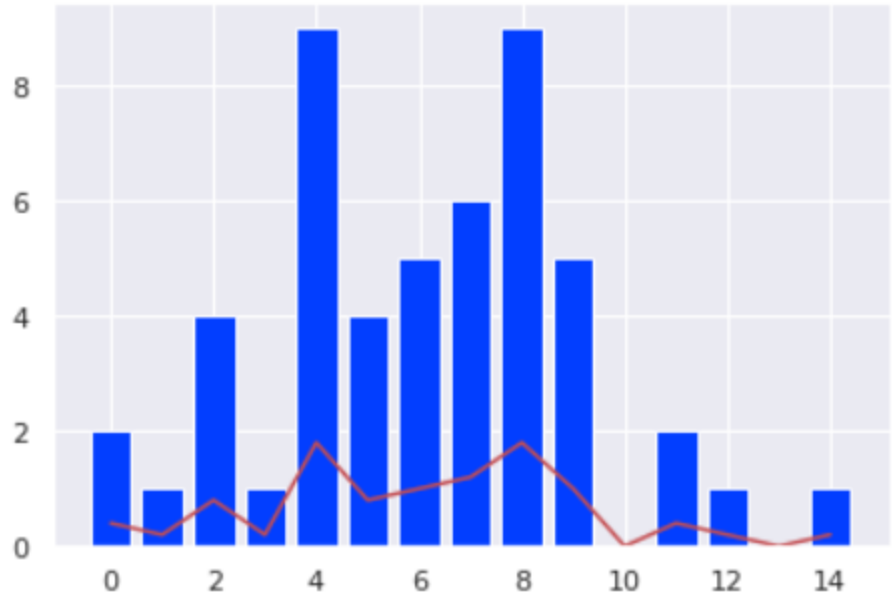
Out[29]: <matplotlib.collections.PathCollection at 0x7faedb76d190>



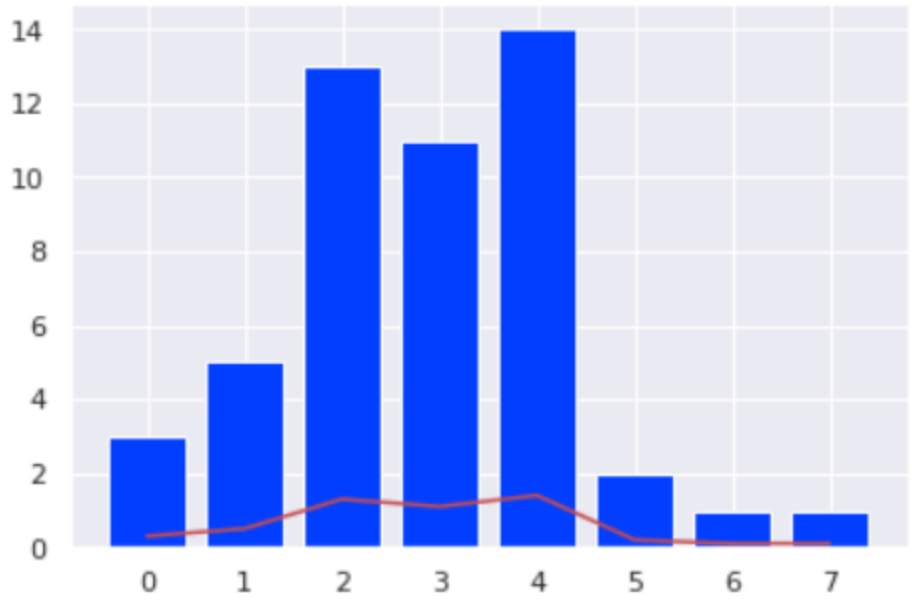
In [32]:
```
display_histogram_density(data_1D, 0.05)
```



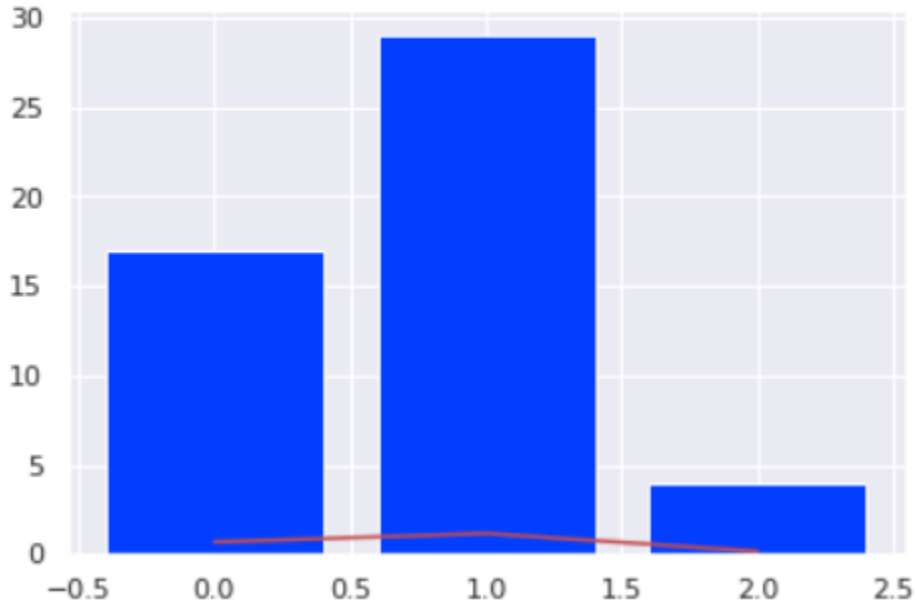
In [33]:
```
display_histogram_density(data_1D, 0.1)
```



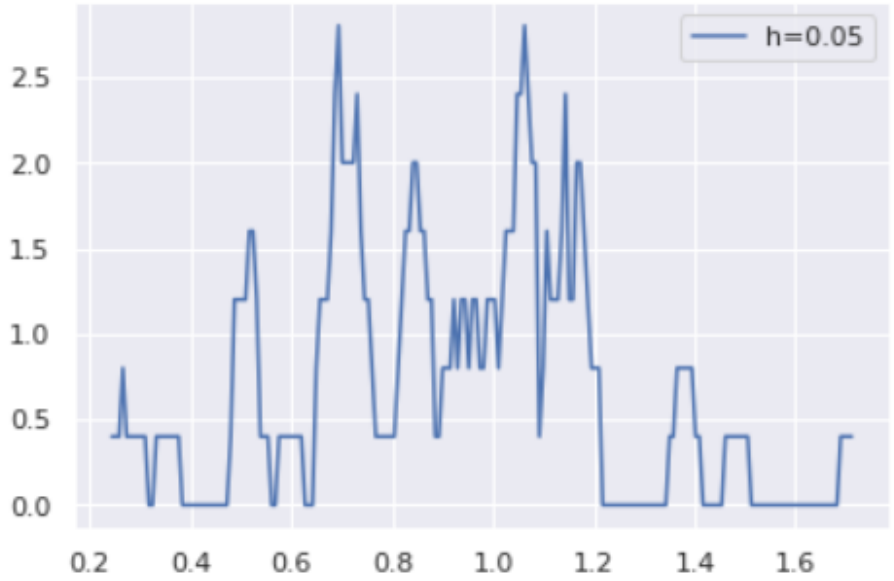
In [34]:
```
display_histogram_density(data_1D, 0.2)
```



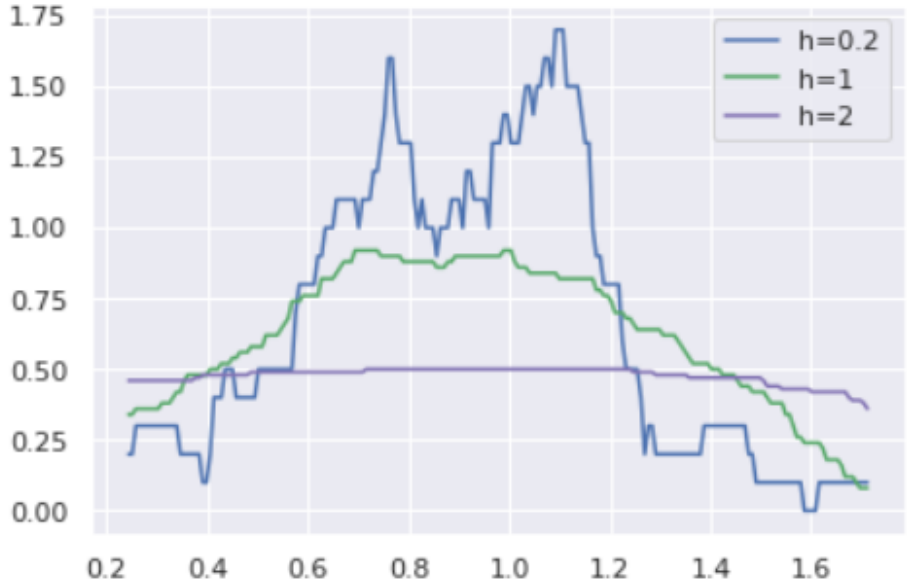
In [35]:
```
display_histogram_density(data_1D, 0.5)
```



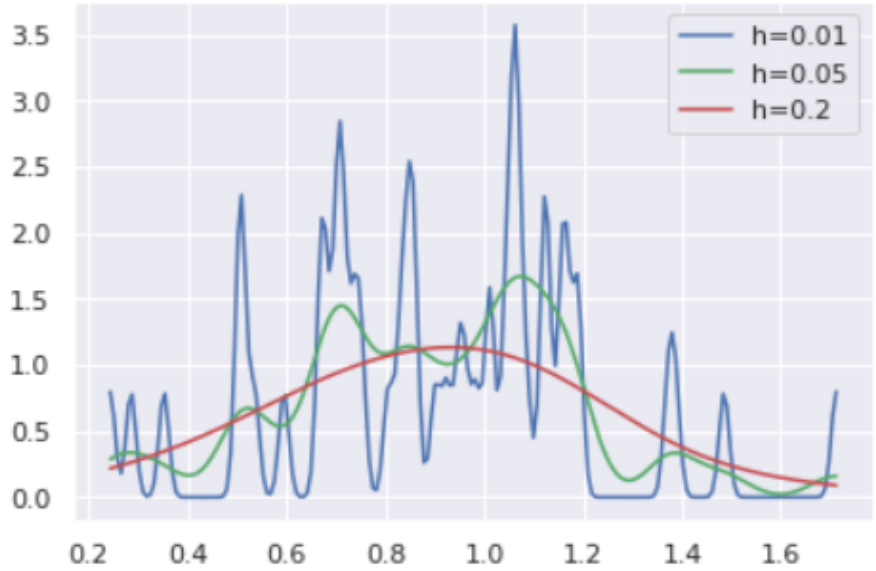
In [39]:
```
display_hypercube_kernel_density_1D(data_1D, 0.05, 'b')
```



In [40]:
```
display_hypercube_kernel_density_1D(data_1D, 0.2, 'b')
display_hypercube_kernel_density_1D(data_1D, 1, 'g')
display_hypercube_kernel_density_1D(data_1D, 2, 'm')
```



In [44]:
```
display_gaussian_kernel_density_1D(data_1D, 0.01, 'b')
display_gaussian_kernel_density_1D(data_1D, 0.05, 'g')
display_gaussian_kernel_density_1D(data_1D, 0.2, 'r')
```



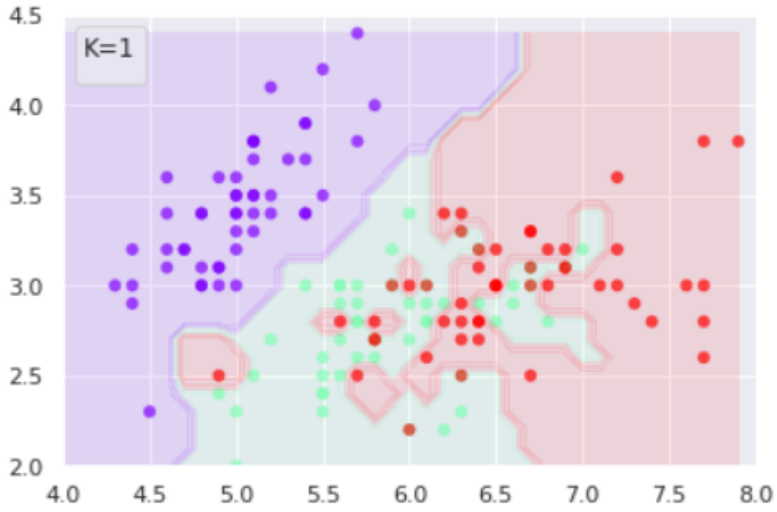
3.1) Generate Data

In [45]:
```
iris = datasets.load_iris()
iris_x = np.array(iris.data[:, :2])  # we only take the firs
iris_t = np.array(iris.target)

def plot_iris(legend=True, classes=iris_t, target=plt):
    scatter = target.scatter(iris_x[:, 0], iris_x[:, 1], c=c
    if legend:
        legend = target.legend(*scatter.legend_elements(), l
        return (scatter, legend)
    return (scatter, )

plot_iris()
```

Out[45]:
```
(<matplotlib.collections.PathCollection at 0x7faedb78b850>,
 <matplotlib.legend.Legend at 0x7faedb879510>)
```

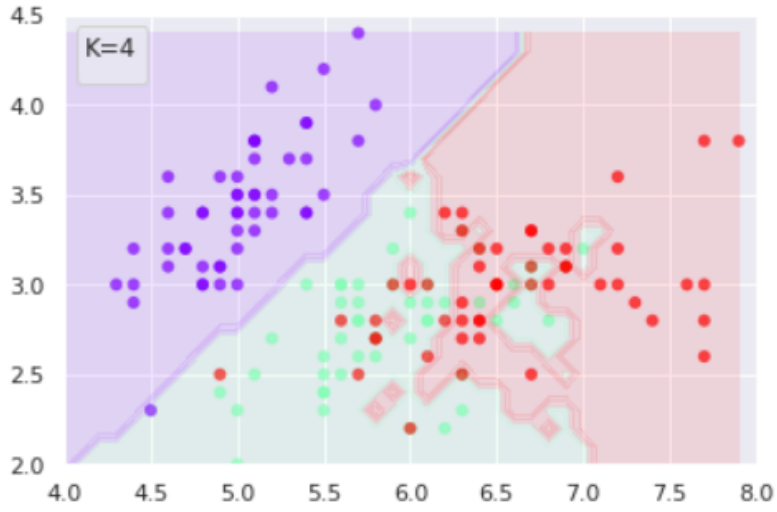
In [48]:
```
plot_iris(False)
plot_mesh(lambda x: k_nearest_classification(x, iris_x, iris_t, 1))
plt.legend([], loc="upper left", title="K=1")
```

Out[48]: <matplotlib.legend.Legend at 0x7faedb88b9d0>



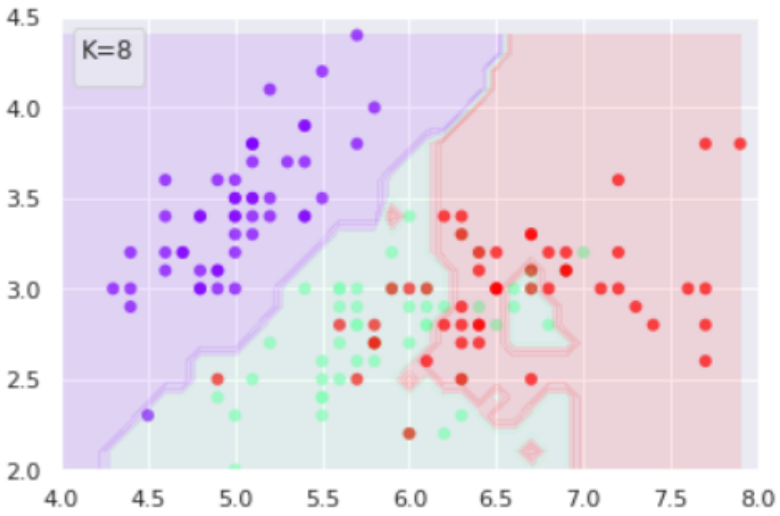
In [49]:
```
plot_iris(False)
plot_mesh(lambda x: k_nearest_classification(x, iris_x, iris_t, 4))
plt.legend([], loc="upper left", title="K=4")
```

Out[49]: <matplotlib.legend.Legend at 0x7faedb5e22d0>



```
In [50]: plot_iris(False)
         plot_mesh(lambda x: k_nearest_classification(x, iris_x, iris_t, 8))
         plt.legend([], loc="upper left", title="K=8")
```

```
Out[50]: <matplotlib.legend.Legend at 0x7faedb63af90>
```



This one will also have a video

In [56]:
```
def create_animation(all_steps, data_x):

    distortions = list(map(
        lambda a: distortion_measure(a[0], a[1], data_x),
        all_steps
    ))

    fig, (ax, ax2) = plt.subplots(1, 2, figsize=(15,5))

    def animate(i):
        ax.cla()
        ax2.cla()

        plot1 = plot_k_means(all_steps[i][0], all_steps[i][1], target=ax)
        ax2.plot(list(range(i)), distortions[:i], '-o')
        plt.xlabel('Step')
        plt.ylabel('Distortion')
        return plot1

    anim = FuncAnimation(
        fig, animate,
        frames=len(all_steps), interval=500, blit=True
    )
    return HTML(anim.to_html5_video())

create_animation(all_steps, iris_x)
```

Out[56]: