//

Machine Learning VI

maXbox Starter 68 – Data Science with Max

What is another word for a python ?
A mega-bite !

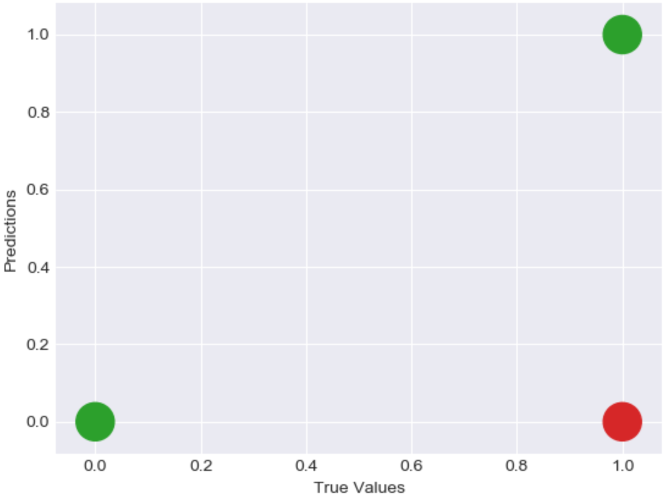
This tutor makes a last step with our classifiers in Scikit-Learn of the preceding tutorial 65, 66 and 67 with clustering and 3D plot.

Lets start with a short repetition, we tested a score with a test set:

```
svm.fit(X_train,y_train)
y_pred = svm.predict(X_test)
print('Class test Score:', svm.score(X_test, y_test))
print(confusion_matrix(y_test, y_pred))
```

```
[[1  0]
 [1  1]]
```

and plotted the false negative:



The true value is one but gets classified (predicted) as zero!

```
class test report:
              precision    recall  f1-score   support

    class 0       0.50      1.00      0.67         1
    class 1       1.00      0.50      0.67         2

avg / total       0.83      0.67      0.67         3
```

Next we want to cluster our dataset. In an unsupervised method such as K Means clustering the target (y) variable is not used in the training process.

```
from sklearn.cluster import KMeans
kmean= KMeans(n_clusters=2,init='k-means++',max_iter=100,
                                        n_init=1, random_state=15)
kmean.fit(X)
```

First we create the kmean model and specify the number of clusters it should find (n_clusters=2) and then we fit the model to the data.
As you can see there's no target needed.
Next we can view the results:

```
print('kmean.clusters \n',np.unique(kmean.labels_, return_counts=True))
print('kmeanclusters thinks: \n',kmean.labels_)
```

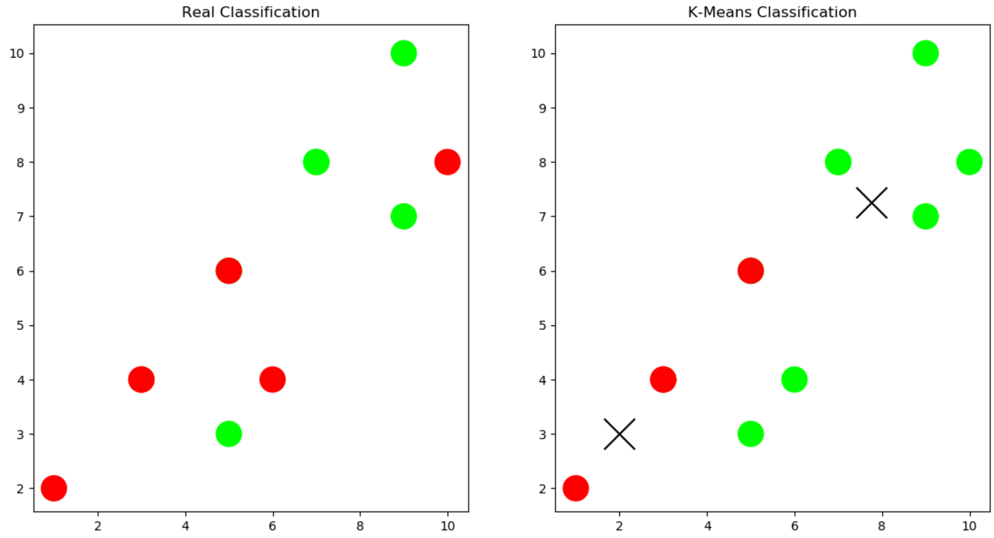
```
kmean.clusters
 (array([0, 1]), array([2, 4], dtype=int64))
kmeanclusters thinks:
 [0 0 1 1 1 1]
```

Not bad for the first try as random state = 15, cause our target is

```
 [0 0 1 1 0 1]
```

```
    A   B   C   D
[[ 1.  2.  3.  4.  0.]
 [ 3.  4.  5.  6.  0.]
 [ 5.  6.  7.  8.  1.]
 [ 7.  8.  9. 10.  1.]
 [10.  8.  6.  4.  0.]
 [ 9.  7.  5.  3.  1.]]
```

Now we plot the model (400 is just the marker bubble size):



The colors mean red for zero and green for one as class. Because the model is unsupervised it did not know which label (class 0 or 1) to assign to each class but we can draw the "centroid" of the clusters.
As you can see an overlay draws all 4 features A,C as X and B,D as Y axis[1] on the plot but you can see only 9 instead of 12 so 3 number pairs are duplicates!

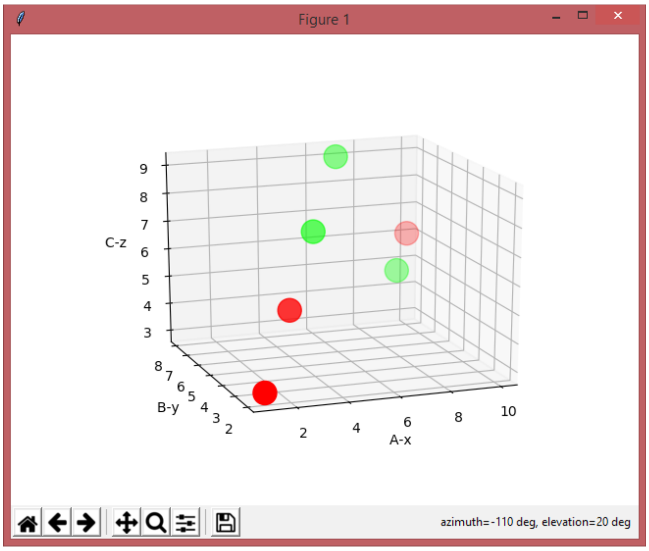
```
plt.subplot(1, 2, 1)
plt.scatter(df.A, df.B, c=colormap[y2.Targets], s=400)
plt.title('Real Classification')
# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(df.A, df.B, c=colormap[kmean.labels_], s=400)
plt.title('K-Mean Classification')
```

[1] We map sort of a 4Dim to a 2Dim plot

```
plt.scatter(centroids[:, 0], centroids[:, 1],
            marker='x', s=600, linewidths=50,
            color='black', zorder=10)

plt.subplot(1, 2, 1)
plt.scatter(df.C, df.D, c=colormap[y2.Targets], s=400)
plt.title('Real Classification')
# Plot the Models Classifications
plt.subplot(1, 2, 2)
plt.scatter(df.C, df.D, c=colormap[kmean.labels_], s=400)
plt.title('K-Means Classification')
plt.show()

plot3D()
```

Ok., our last step is to plot the data set in 3D.
The idea of 3D scatter plots is that you can compare 3 characteristics of a data set instead of two. How to build a 3D diagram in Python from the ground up.

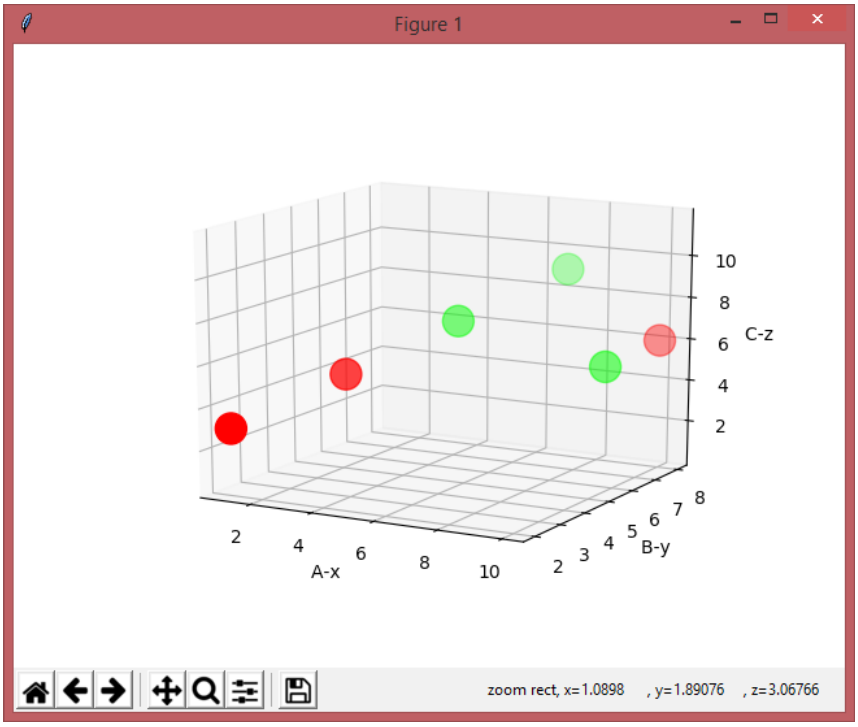
```
from mpl_toolkits.mplot3d import Axes3D

def plot3D():
    colormap = np.array(['red', 'lime'])
    targets2 = [0,0,1,1,0,1]
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    x =[1,3,5,7,10,9]
    y =[2,4,6,8,8,7]
    z =[3,5,7,9,6,5]
    ax.scatter(x, y, z, c = colormap[targets2], marker='o', s=300)
    ax.set_xlabel('A-x ')
    ax.set_ylabel('B-y ')
    ax.set_zlabel('C-z ')

    plt.show()
```



We set A, B and C as the x, y and z-axis, but not the red one D as depicted:

```
   A    B    C    D
[[ 1.   2.   3.   4.]
 [ 3.   4.   5.   6.]
 [ 5.   6.   7.   8.]
 [ 7.   8.   9.  10.]
 [10.   8.   6.   4.]
 [ 9.   7.   5.   3.]]
```

Besides 3D wires and printing in 3D, planes, one of the most popular 3-dimensional graph types is 3D scatter plots.



zoom rect, x=1.0898 , y=1.89076 , z=3.06766

As you can see the first sample of class 0 has the coordinates ~ 1,2,3 in the [zoom rect] above in the picture.
Lets take the features A,B and C to plot that generates a basic 3D scatter plot that goes with a X, Y, Z matrix:

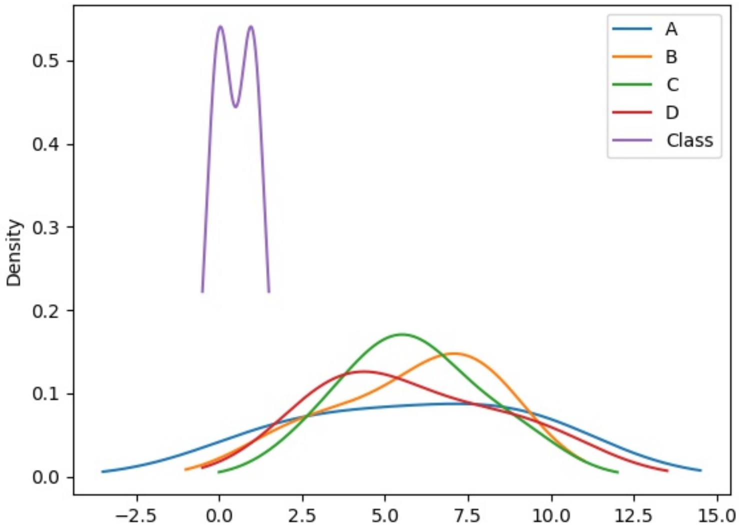
```
                X  Y  Z    Class
Dataset   = [[1, 2, 3,      0],
             [3, 4, 5,      0],
             [5, 6, 7,      1],
             [7, 8, 9,      1],
             [10,8, 6,      0],
             [9, 7, 5,      1]]
```

3D plots are also enabled by importing the mplot3d toolkit, included with the main Matplotlib installation: `from mpl_toolkits import mplot3d`
Note that by default, the scatter points have their transparency adjusted to give a sense of depth on the page[2].

Mathematically, a histogram is a mapping of bins (intervals or numbers) to frequencies. More technically, it can be used to approximate a probability density function (PDF) of the underlying variable that we see later on.
Moving on from a frequency table above (density=**False** counts at y-axis), a true histogram first <bins> the range of values and then counts the number of values that fall into each bin or interval. A plot of a histogram uses its bin edges on the x-axis and the corresponding frequencies on the y-axis.

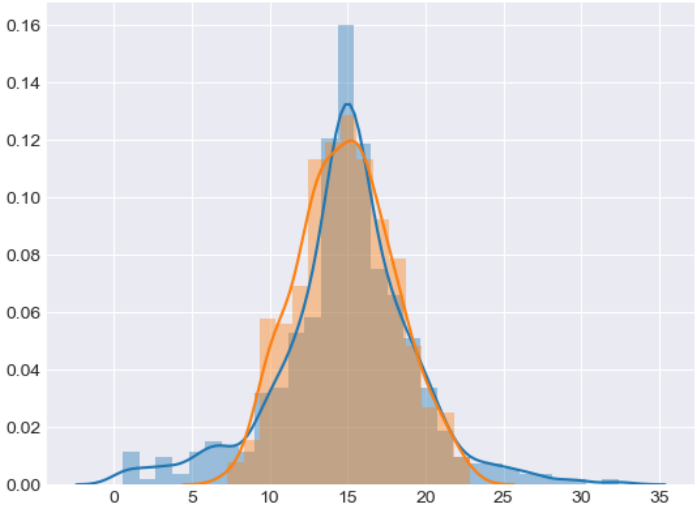
Sticking with the Pandas library, you can create and overlay density plots using plot.kde(), which is available for both [Series] and [DataFrame] objects.
`df.iloc[0:,0:4].plot.kde()`

[2] https://jakevdp.github.io/PythonDataScienceHandbook/04.12-three-dimensional-plotting.html



This is also possible for our binary targets to see a probabilistic distribution of the target class values (labels in supervised learning): **[0. 0. 1. 1. 0. 1.]** Consider at last a sample of floats drawn from the Laplace and Normal distribution together. This distribution graph has fatter tails than a normal distribution and has two descriptive parameters (location and scale):

```
>>> d = np.random.laplace(loc=15, scale=3, size=500)
>>> d = np.random.normal(loc=15, scale=3, size=500)
```



The script can be found:
http://www.softwareschule.ch/examples/classifier_compare2confusion2.py.txt

Ref:
http://www.softwareschule.ch/box.htm
https://scikit-learn.org/stable/modules/
https://realpython.com/python-histograms/
Doc:
https://maxbox4.wordpress.com