UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Centro de e-Learning

UNIDAD DIDÁCTICA VIII

MACHINE LEARNING

Centro de e-Learning SCEU UTN - BA.
Medrano 951 2do piso (1179) // Tel. +54 11 4867 7589 / Fax +54 11 4032 0148
www.sceu.frba.utn.edu.ar/e-learning

Módulo II – Herramientas avanzadas y casos de estudio

Unidad VIII – sklearn – No Supervisado II

Presentación:

En esta unidad seguiremos utilizando sklearn para analizar dos tipos de modelos no supervisados e introduciremos el concepto de estimadores de densidad, que nos permiten visualizar la distribución de probabilidad aproximada de la cual provienen nuestros datos.

Además introduciremos el análisis de componentes principales como herramienta para la reducción de dimensionalidad de problemas complejos.

Objetivos:

Que los participantes:

Aprendan a trabajar con modelos no supervisados.

Afiancen el proceso de creación de modelo con sklearn y la visualización de datos.

Bloques temáticos:

1.- Estimadores de densidad.

2.- Análisis de componentes principales (PCA).

Consignas para el aprendizaje colaborativo

En esta Unidad los participantes se encontrarán con diferentes tipos de actividades que, en el marco de los fundamentos del MEC*, los referenciarán a tres comunidades de aprendizaje, que pondremos en funcionamiento en esta instancia de formación, a los efectos de aprovecharlas pedagógicamente:

- Los foros proactivos asociados a cada una de las unidades.
- La Web 2.0.
- Los contextos de desempeño de los participantes.

Es importante que todos los participantes realicen algunas de las actividades sugeridas y compartan en los foros los resultados obtenidos.

Además, también se propondrán reflexiones, notas especiales y vinculaciones a bibliografía y sitios web.

El carácter constructivista y colaborativo del MEC nos exige que todas las actividades realizadas por los participantes sean compartidas en los foros.

Tomen nota

Las actividades son opcionales y pueden realizarse en forma individual, pero siempre es deseable que se las realice en equipo, con la finalidad de estimular y favorecer el trabajo colaborativo y el aprendizaje entre pares. Tenga en cuenta que, si bien las actividades son opcionales, su realización es de vital importancia para el logro de los objetivos de aprendizaje de esta instancia de formación. Si su tiempo no le permite realizar todas las actividades, por lo menos realice alguna, es fundamental que lo haga. Si cada uno de los participantes realiza alguna, el foro, que es una instancia clave en este tipo de cursos, tendrá una actividad muy enriquecedora.

Asimismo, también tengan en cuenta cuando trabajen en la Web, que en ella hay de todo, cosas excelentes, muy buenas, buenas, regulares, malas y muy malas. Por eso, es necesario aplicar filtros críticos para que las investigaciones y búsquedas se encaminen a la excelencia. Si tienen dudas con alguno de los datos recolectados, no dejen de consultar al profesor-tutor. También aprovechen en el foro proactivo las opiniones de sus compañeros de curso y colegas.

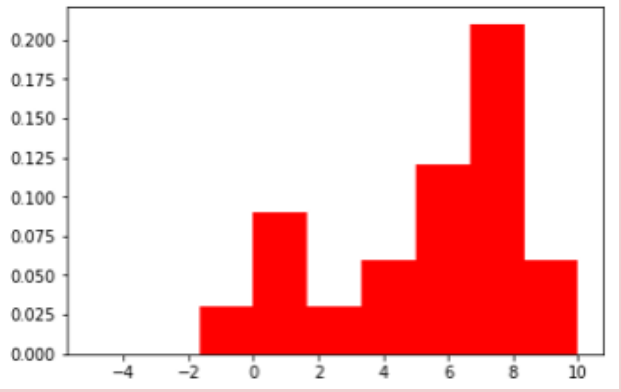
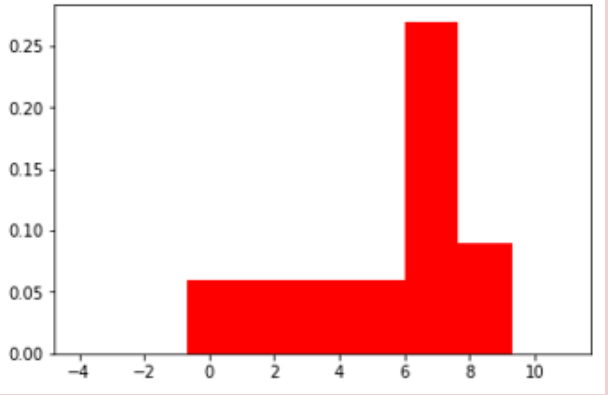
1. Estimador de densidad.

La estimación de la densidad recorre la línea entre el aprendizaje no supervisado, la ingeniería de características y el modelado de datos, es un concepto muy simple, y la mayoría de las personas ya están familiarizadas con los gráficos de histogramas, los cuales son una técnica común de estimación de densidad. Un histograma es una visualización de datos donde se definen los contenedores (rangos de valores agrupados), y se cuenta la cantidad de puntos de datos dentro de cada contenedor.

Como ejemplo podemos considerar el siguiente caso en el cual se crea un vector “X” a partir de concatenar dos distribuciones normales, una centrada en 2 y la otra centrada en 7 con la misma desviación standard igual a 1 y diferente cantidad de puntos.

```
import numpy as np
import matplotlib
import matplotlib.pyplot as plt
from scipy.stats import norm
np.random.seed(1)
X = np.concatenate((np.random.normal(2, 1, 6),  np.random.normal(7, 1, 14)))
X =X[:, np.newaxis]
numero_intervalos= np.linspace(-5, 10, 10)  #Número de intervalos
plt.hist(X[:, 0], numero_intervalos, facecolor = 'red', **{'density': True})
plt.show()
```

Podemos ver que el gráfico obtenido puede variar mucho con la selección de los intervalos (numero_intervalos)

numero_intervalos = np.linspace(-5, 10, 10)	numero_intervalos +1
	

La representación de datos mediante histogramas como podemos notar, presenta un problema importante, ya que la elección de los contenedores puede tener un efecto desproporcionado en la visualización resultante, por lo que en lugar de considerar agrupar rangos de valores, sería mejor centrar cada dato en el punto que representa y sumar las alturas totales de cada ubicación. De esta forma podemos obtener un modelo no paramétrico de la distribución de puntos.

Sklearn utiliza el estimador de densidad "**sklearn.neighbors.KernelDensity**" para realizar esta tarea, implementando internamente una función positiva llamada "**Kernel**" que se encuentra controlada por el ancho de banda "**h**" a partir de una serie de datos **x**. La función kernel la podríamos representar de la siguiente manera: $K(x, h)$

Y el estimador de densidad en un punto "**y**" dentro de un grupo "**x**" de esta forma:

$$\rho_K(y) = \sum_{i=1}^{N} K\left(\frac{(y - x_i)}{h}\right)$$

Las diferentes funciones **K** con las que podemos contar son:

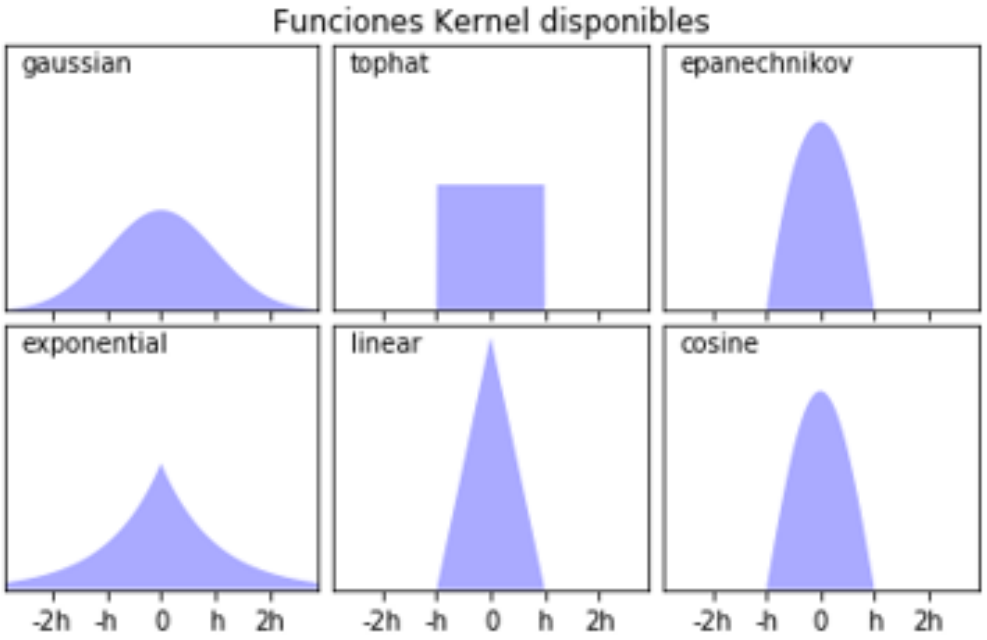
- 'gaussian',
- 'tophat',
- 'epanechnikov',
- 'exponential',
- 'linear',
- 'cosine'

Podemos realizar una representación gráfica de las mismas de la siguiente forma:

```
X_plot = np.linspace(-6, 6, 1000)[:, None]
X_src = np.zeros((1, 1))

fig, ax = plt.subplots(2, 3, sharex=True, sharey=True)
fig.subplots_adjust(left=0.05, right=0.95, hspace=0.05, wspace=0.05)


def format_func(x, loc):
    if x == 0:
        return '0'
    elif x == 1:
```

```
        return 'h'
    elif x == -1:
        return '-h'
    else:
        return '%ih' % x

for i, kernel in enumerate(['gaussian', 'tophat', 'epanechnikov', 'exponential', 'linear', 'cosine']):
    axi = ax.ravel()[i]
    log_dens = KernelDensity(kernel=kernel).fit(X_src).score_samples(X_plot)
    axi.fill(X_plot[:, 0], np.exp(log_dens), '-k', fc='#AAAAFF')
    axi.text(-2.6, 0.95, kernel)

    axi.xaxis.set_major_formatter(plt.FuncFormatter(format_func))
    axi.xaxis.set_major_locator(plt.MultipleLocator(1))
    axi.yaxis.set_major_locator(plt.NullLocator())

    axi.set_ylim(0, 1.05)
    axi.set_xlim(-2.9, 2.9)

ax[0, 1].set_title('Funciones Kernel disponibles')
plt.show()
```

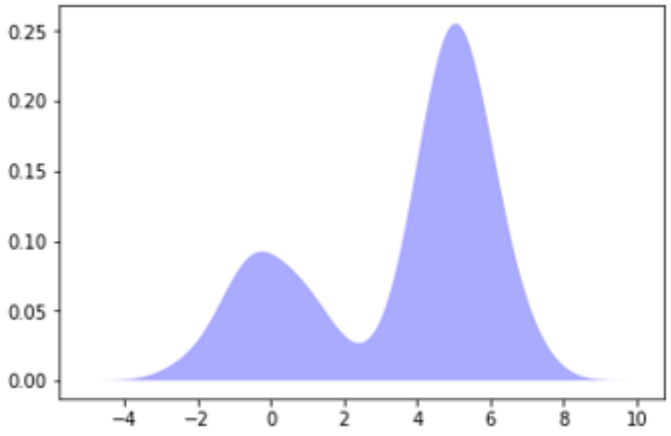
Retornando:



Si ahora realizamos un ajuste de los datos tomando por ejemplo la distribución gausiana:

```
from sklearn.neighbors import KernelDensity

kde = KernelDensity(kernel='gaussian', bandwidth=0.75).fit(X)
log_dens = kde.score_samples(X_plot)
X_plot = np.linspace(-5, 10, 1000)[:, np.newaxis]
plt.fill(X_plot[:, 0], np.exp(log_dens), fc='#AAAAFF')

plt.show()
```

En lugar de nuestro histograma podemos obtener:



2. Analisis de componentes principales (PCA)

El procesamiento de datos se utiliza para descomponer un conjunto de datos con múltiples características en un conjunto de componentes ortogonales sucesivos que explican una cantidad máxima de la varianza. En scikit-learn, PCA se implementa como un objeto transformador que aprende componentes en su método de ajuste, y se pueden utilizar en nuevos datos para proyectarlos en estos componentes.

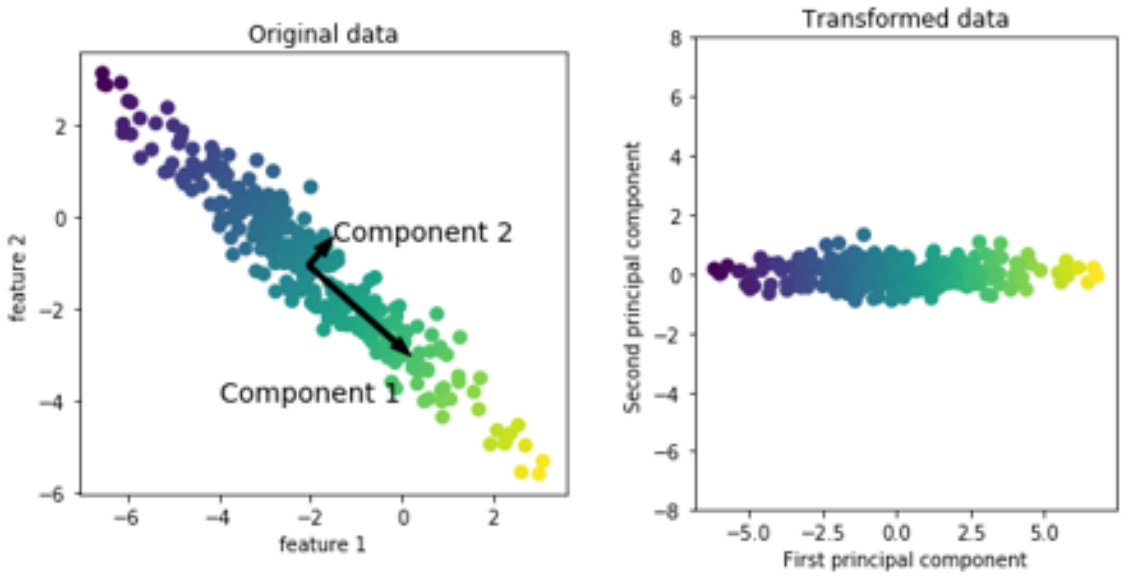
Podemos ver cómo trabaja, si cargamos el siguiente módulo:

```
import mglearn
```

Y ejecutamos:

```
mglearn.plots.plot_pca_illustration()
```

Esto nos despliega una serie de gráficos, en donde en el primero se muestra una representación en dos dimensiones de dos características (feature2 vs feature1). En el gráfico vemos cómo a partir de la muestra, **pca** toma la dirección en donde existe una mayor variación de datos (mayor varianza) y traza un eje que denomina “componente 1” el cual es tomado como el componente principal, y luego toma el eje ortogonal (90°) y traza el “componente 2”, finalmente muestra una representación de los ejes sobre la distribución de puntos. Ambos ejes se trazan de forma tal de maximizar la varianza.

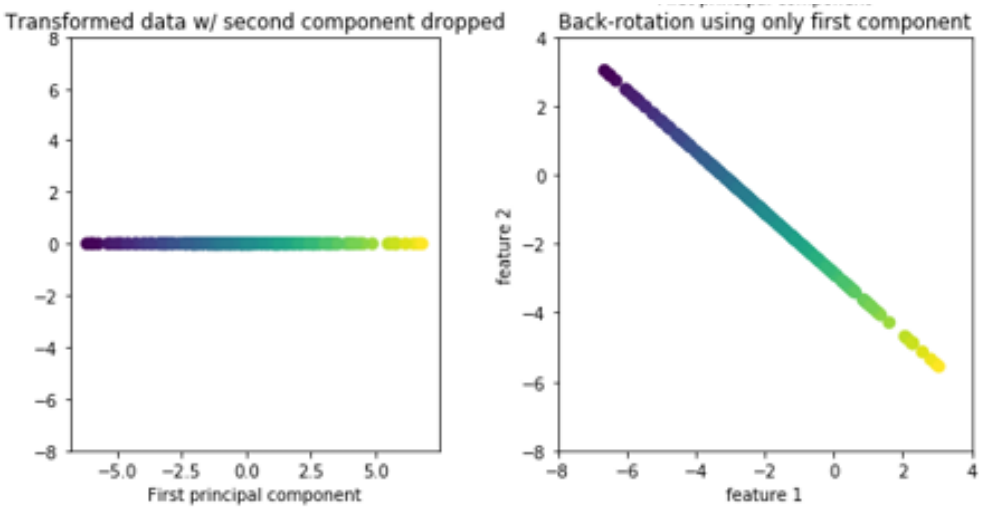


A partir de los dos ejes hallados intenta graficar todos los puntos y realiza una representación abstracta de los datos. En los gráficos anteriores se puede ver:

- En el **gráfico 1**, el trazado de los nuevos ejes.
- En el **gráfico 2**, la nueva asignación de ejes, lo cual causa una rotación de la representación, en donde a cada valor se le ha restado el promedio de datos para centrar los valores en cero tanto en el eje x como en el eje y.

También nos muestra un par de gráficos más en donde podemos ver:

- En el **gráfico 3**, la proyección de los datos sobre el nuevo eje correspondiente al componente principal.
- En el **gráfico 4**, elimina el ruido de los datos, el gráfico se ve con la misma dirección al gráfico 1 pero representado como una línea recta.



Utilizamos pca para reducir el número de dimensiones que tenemos en el set de datos, indicando que realice la representación en dos ejes, y luego le cargamos los datos de iris.

```
pca=PCA(n_components=2)
pca.fit(iris.data)
```

Podemos ver que nuestros datos originales parten con un set de 150 datos y cuatro características y que luego tenemos 150 datos y dos caracterisitcas:

```
print(iris.data.shape)
print(transformada.shape)
```

Resultado:

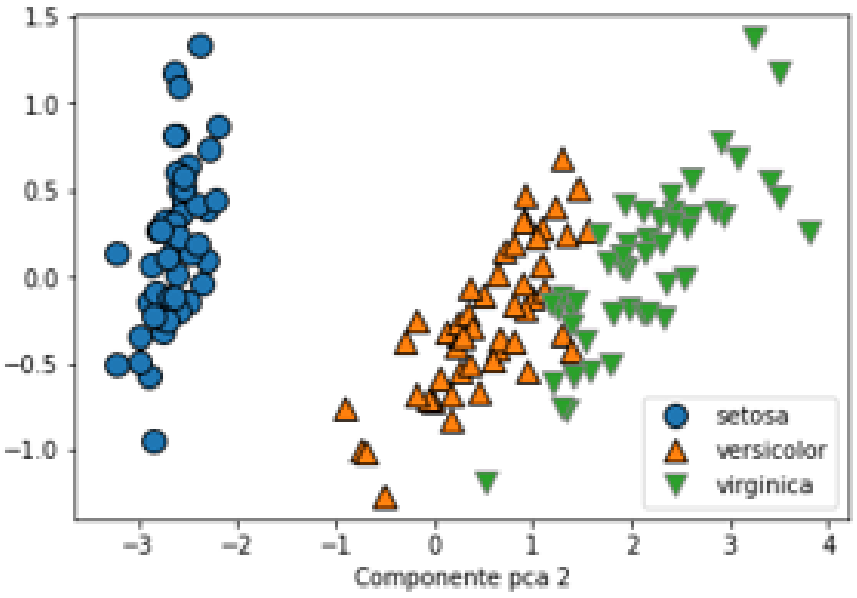
(150, 4)

(150, 2)

Al ejecutar

```
mglearn.discrete_scatter(transformada[:,0], transformada[:,1], iris.target)
plt.legend(iris.target_names, loc='lower right')
plt.xlabel('Componente pca 1')
plt.ylabel('Componente pca 2')
```

Podemos ver como el modelo realiza una clasificación de los datos en tres grupos.

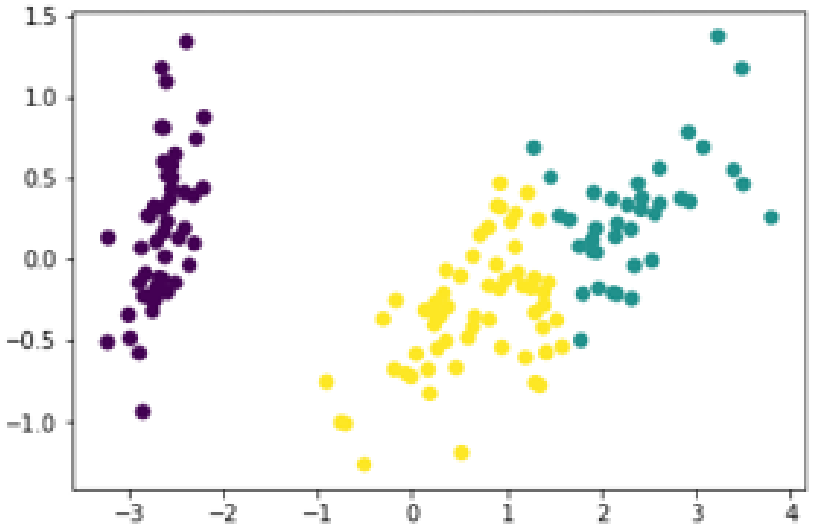


Podríamos seguir la recomendación de tomar estos datos como entrada de K-means y ver el resultado obtenido:

```
k_means = KMeans(n_clusters=3, max_iter=2000)
k_means.fit(transformada)
 predicciones=k_means.predict(transformada)
 score=metrics.adjusted_rand_score(etiquetas, predicciones)
print(score)
plt.scatter(transformada[:, 0], transformada[:, 1], c=predicciones)
```

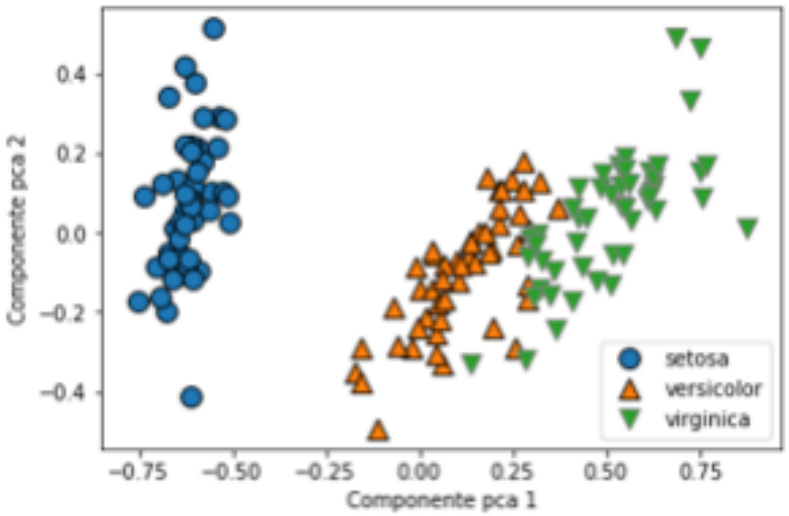
```
plt.show()
```

El resultado nos muestra un 71,6% de aciertos, no evidenciándose en este caso una mejora en el método de predicción.



Si ahora aplicamos el proceso de escalado que vimos en la clase pasada a nuestro trabajo, obtenemos:

```
from sklearn.preprocessing import MinMaxScaler
escala=MinMaxScaler()
escala.fit(iris.data)
escalada=escala.transform(iris.data)
pca.fit(escalada)
transformada=pca.transform(escalada)
mglearn.discrete_scatter(transformada[:,0], transformada[:,1], iris.target)
plt.legend(iris.target_names, loc='lower right')

plt.xlabel('Componente pca 1')
plt.ylabel('Componente pca 2')
```



En el caso de la librería iris no existe una diferencia apreciable en la representación de datos. Esto puede variar considerablemente en otros casos en donde la variación de valores es más apreciable.

Bibliografía utilizada y sugerida

Libros

Programming Python 5th Edition – Mark Lutz – O'Reilly 2013

Mastering Exploratory Analysis with pandas - By Harish Garg - September 2018

Manual online

(2019, 01). Obtenido 01, 2019, de https://pandas.pydata.org/

(2019, 01). Obtenido 01, 2019, de https://matplotlib.org/

(2019, 01). Obtenido 01, 2019, de https://scikit-learn.org/stable/

Lo que vimos

En esta unidad hemos trabajado estimadores de densidad, y con análisis de componentes principales.