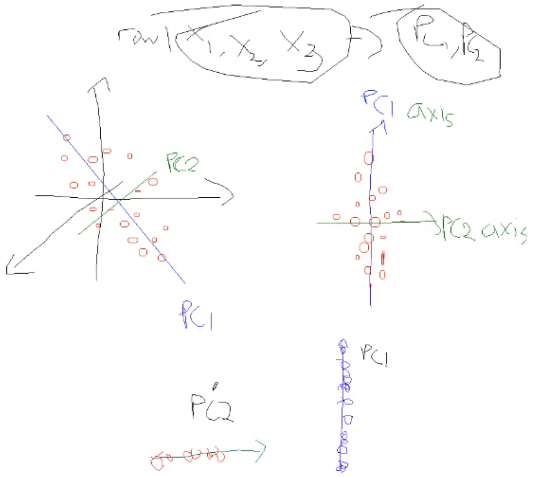
Nama : Ghina Khoerunnisa
Program : Data Scientist Intern

Unsupervised Learning

Unsupervised learning tidak “dipandu” dengan labelnya seperti di supervised learning. Unsupervised learning bertujuan untuk eksplorasi, dimensionality reduction, extract, detect pattern, dan dapat meningkatkan performa. Dua kasus penggunaan umum untuk unsupervised learning adalah exploratory analysis dan dimensionality reduction. "Clustering" adalah istilah yang digunakan untuk menggambarkan eksplorasi data. Unsupervised learning lebih berguna di real world nya karena data di dunia nyatanya itu tidak ada labelnya dan melabelkan dataset manual itu memiliki cost yang tinggi. Dimensionality reduction digunakan karena jika kita punya data yang fiturnya banyak maka kita semakin membutuhkan record yang banyak juga karena variasinya akan besar.

Principal Component Analysis

Principal Component Analysis merupakan ekstraksi informasi dari ruang dimensi tinggi dengan memproyeksikannya ke ruang dimensi yang lebih rendah. Cara kerjanya adalah pertama akan mencari rata-rata variabel dari kumpulan data point yang kita punya, kemudian setelah pemusatan rata-rata dan penskalaan ke varian unit, kumpulan data siap untuk penghitungan summary index pertama, principal component pertama (PC1). Lalu, begitu juga dengan principal component kedua (PC2). Ketika dua principal components ditemukan, mereka bersama-sama menentukan place, window ke ruang variabel K-dimensional. Dengan memproyeksikan semua pengamatan ke sub-ruang berdimensi rendah dan memplot hasilnya.



Implementasi PCA dengan Python (menggunakan scikit-learn) :

```
from sklearn.decomposition import PCA
pca_breast = PCA(n_components=2)
principalComponents_breast = pca_breast.fit_transform(x)
```

```
principalComponents_breast
```

```
array([[ 9.19283683,  1.94858307],
       [ 2.3878018 , -3.76817174],
       [ 5.73389628, -1.0751738 ],
       ...,
       [ 1.25617928, -1.90229671],
       [10.37479406,  1.6720101 ],
       [-5.4752433 , -0.67063679]])
```

K-Means Clustering

Clustering merupakan pengelompokan sekumpulan objek sedemikian rupa sehingga objek dalam kluster yang sama lebih mirip satu sama lain daripada objek di kluster lain. Terdapat dua properti penting di Clustering. Pertama setiap data point di dalam suatu cluster harus similar dengan data point di cluster yang sama (memiliki karakteristik yang sama). Kedua, data point dalam suatu cluster harus seberbeda mungkin dengan data point di cluster lainnya. Untuk mencari nilai cluster yang optimum kita dapat menggunakan elbow method yang menggunakan nilai inertia. Inertia menghitung jumlah jarak semua titik dalam sebuah cluster dari pusat cluster tersebut kemudian menghitung jumlah untuk semua cluster. Nilai K(banyak cluster) yang optimum itu memiliki jarak yang serendah mungkin. Cara kerja dari K-means itu sendiri yaitu pertama generate K random centroid dari data point yang ada, kemudian yang kedua hitung jarak setiap data point ke setiap cluster yang ada lalu assign data point tersebut ke dalam cluster dengan centroid terdekatnya. Kemudian perbarui centroid dengan mengassignnya ke tengah-tengah cluster, lalu ulang terus step kedua hingga memperbarui centroid hingga centroidnya tidak berpindah-pindah lagi/setiap data point berada di cluster yang sama atau tidak berpindah-pindah cluster lagi. Aplikasi dari K-means clustering ini adalah customer profiling, market segmentation, computer vision dan lain-lain.

Implementasi K-means Clustering dengan Python (menggunakan scikit-learn) :

Import K-means,

```
1 from sklearn.cluster import KMeans
```

Find K value,

```
wcss = []
cluster = {}
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', max_iter = 300, n_init = 10, random_state = 0)
    kmeans.fit(X)
    cluster[i] = kmeans
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss)
plt.title('The elbow method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS') #within cluster sum of squares
plt.show()
```

Cluster Model,

```
model_kmean = cluster[2]
model_kmean.fit(X)
```

```
KMeans(n_clusters=2, random_state=0)
```

```
clst = model_kmean.cluster_centers_
```

```
array([[-0.99958666, -0.8700527 ],
       [ 2.10591542,  2.03938883]])
```

Note :

Clustering dengan fitur yang banyak dapat divisualisasikan dengan cara,

- **Lakukan PCA terlebih dahulu kemudian clustering lalu visualize**
- **Clustering terlebih dahulu kemudian PCA lalu visualize**