**PCA - Principal Component Analysis**

* PCA ca procedură Dimensionality Reduction
* PSA – transformarea caracteristicilor setului de date în alte caracteristici independente una de alta numite Principal Component (PC) care sunt sortate în funcție de cantitate de informație conținută
* Necesitatea reducerii dimensiunilor: vizualizare mai simplă, timp mai mic de prelucrare ulterioară
* Utilizarea dispersiei ca parametru de determinare a importantei caracteristicilor datelor nesupravegheate
* Crearea PC1 și PC2 pe baza datelor bidimensionale cu ajutorul dispersiei
* Numărul maxim PC va coincide cu numărul de caracteristici dar informația de bază se va conține în primele PC
* Reducerea dimensiunilor constă în excluderea ultimelor PC în care se conține un procentaj foarte mic de informație
* Noțiunea de scalarea standard permite modificarea scării caracteristicilor datelor prin plasarea punctului mediu a datelor în punctul 0 a sistemului de coordonate, iar valorile noi a datelor se vor determina cu scorul z ***z=(valoare date – valoare medie)/abatere standard***
* Transformarea liniară a datelor presupune înmulțire dot a matricei coordonatelor punctului datelor scalate cu o matrice pătratică si obținerea unei alte matrice de coordonate a unui punct într-un alt plan
* EigenVector – vectorii până la transformare și după transformare care pornesc din origine și care se notează ca matrice cu o coloana si valorile ce corespund coordonatelor vârfului
* EigenValue – amplitudinea vectorilor până la transformare și după transformare care pornesc din origine
* EigenVector ortogonal este un vector perpendicular vectorului EivenVector
* Matricea covarianțelor este o matrice pătratică de dimensiunile numărului de caracteristici care pe diagonala principală conține dispersiile caracteristicilor iar pe celelalte poziții covarianța dintre caracteristici amplasate simetric față de diagonala principală
* Etapele algoritmului PCA:
* Scalarea standard a datelor și deplasarea centrului datelor în punctul de intersecție a coordonatelor caracteristicilor
* Determinarea matricei covarianțelor dintre caracteristici
* Transformarea liniară a datelor prin înmulțirea dot a coordonatelor punctelor cu matricea covarianțelor
* Determinarea vectorilor EigenVector și EigenValue
* Selectarea Eigen vectorilor cu cele mai mare valori a EigenValue în funcție de numarul de PC dorit
* Proiectarea tuturor datelor pe vectorii Eigen selectați si formarea PC
* Implementarea manuală a algoritmului PCA în Python
* Importul datelor
* Elaborarea dependentei heatmap a datelor ***sns.heatmap(df);***
* Importul modulului StandardScaler ***from sklearn.preprocessing import StandardScaler***
* Crearea unui obiect de scalarea ***scaler = StandardScaler()***
* Scalarea datelor df ***df\_scalat = scaler.fit\_transform(df)***
* Determinarea matricei convariantei ***covariance\_matrix = np.cov(df\_scalat, rowvar=False)***
* Determinarea vectorilor Eigen si valorilor Eigen ***eigen\_values,eigen\_vectors=np.linalg.eig(covariance\_matrix)***
* selectarea numarului de PC ce nu vor fi excluse ***num\_componente=2***
* determinarea si aranjarea indexilor compontelor in ordine descrescatoare a valorilor eigen ***index\_componente = np.argsort(eigen\_values)[::-1]***
* selectarea doar a indexilor primelor componente ***index\_primele=index\_componente[:num\_componente]***
* obtinerea celor mai mari valori eigen ***eigen\_values\_max= eigen\_values[index\_primele]***
* obtinerea vectorilor engen cu cea mai mare valoarea eigen ***eigen\_vector\_max=eigen\_vectors[:,index\_primele]***
* proiectarea datelor initaile pe vectorul eigen si determinarea PC ***principal\_components=np.dot(df\_scalat,eigen\_vector\_max)***
* Vizualizarea dependentei PC2 de PC1 ***plt.scatter(principal\_components[:,0],principal\_components[:,1])***
* Etapele algoritmului PCA in Scikit-Learn:
* importul modulului PCA ***from sklearn.decomposition import PCA***
* crearea modelului PCA pentru 2 compnente PC ***pca = PCA(n\_components=2)***
* ajustarea modelului și transformarea datelor df\_scalat ***principal\_components=pca.fit\_transform(scaled\_X)***
* Vizualizarea dependentei dintre PC ***plt.scatter(principal\_components[:,0],principal\_components[:,1])***
* vizualizarea numărului de PC ***pca.n\_components***
* vizualizarea PC ***pca.components\_***
* vizualizarea dispersiei pentru fiecare PC ***pca.explained\_variance\_***
* determinarea raportului de informație continuata in fiecare PC ***pca.explained\_variance\_ratio\_***
* determinarea raportului de informație conținută in primele 2 PC ***np.sum(pca.explained\_variance\_ratio\_)***
* crearea modelului PCA cu toate cele 30 componente PC ***pca\_30 = PCA(n\_components=30)***
* ajustarea modelului pe datele df\_scalat ***pca\_30.fit(scaled\_X)***
* determinarea raportului de informație continuata in fiecare PC ***pca\_30.explained\_variance\_ratio\_***
* determinarea raportului de informație conținută in toate cele 30 PC ***np.sum(pca\_30.explained\_variance\_ratio\_)***
* determinarea raportului de informație conținută in diferite numere de PC ***info***
* afișarea grafica a evoluției raportului de informație in funcție de numărul de PC ***plt.plot(range(1,30),info)***