# Beskrivning

Koden är skapad i VS code med en anaconda inteprenter, pca:conda. Server är skriven i Python och använder Sanic och klientsidan är skriven med React och använder Vite.

Data är från scikit-learn. Viss lagring/hämtning av data via sqlite på serversidan. Beräkning av PCA sker på serversidan.

Start server: with sanic api.app

Start client: with npm run dev

Kopiera conda environment via environment.yml eller requirements.txt

Vid behov kör följande:

**React**

npm install

**Sanic**

python -m pip install sanic

**sqlite**

pip install databases

pip install db-sqlite3

pip3 install aiosqlite

**plot react**

npm install react-plotly.js plotly.js

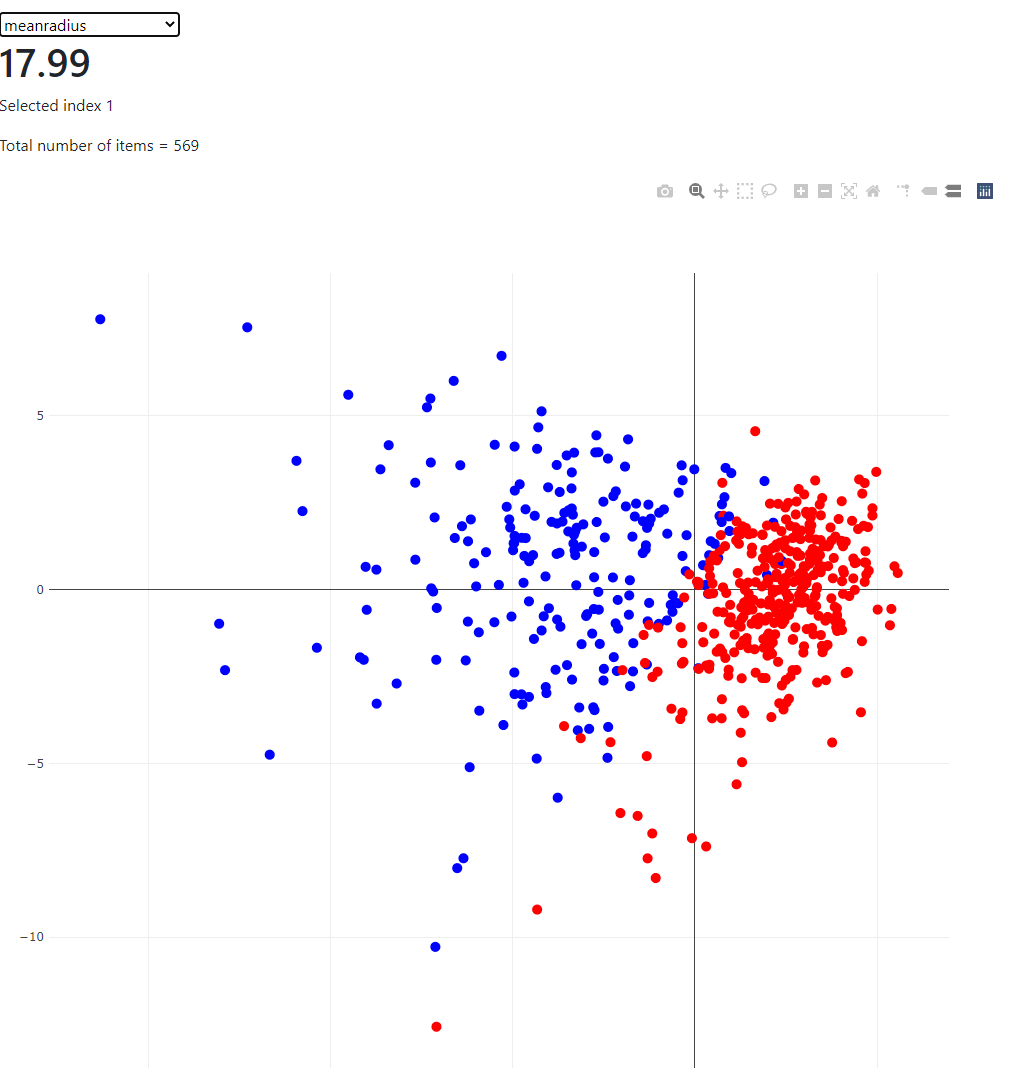
Server: [127.0.0.1:8000/rest](http://127.0.0.1:8000/rest)

[{"route":"/rest","methods":["GET"],"description":"This route: The API documentation"},{"route":"/rest/pcaxy","methods":["GET"],"description":"pcaxy"},{"route":"/rest/patient/:id","methods":["GET"],"description":"patient matching id"},{"route":"/rest/patients","methods":["GET"],"description":"patients"},{"route":"/rest/pcacolumns","methods":["GET"],"description":"pcacolumns"},{"route":"/rest/get\_patientdata/:id/:marker","methods":["GET"],"description":"patient data"},{"route":"/rest/patientclass","methods":["GET"],"description":"patientclass"}]

Klient: <http://localhost:3000/>

# Funktion

En PCA plot visar dataset i två dimensioner med röd färg för Malignant och blå färg för Benign(godartat). Det är möjligt att välja en av 30 markörer(attribute) och få motsvarande värde för den punkten/patienten som användaren klickar på. Tanken är på ett visuellt sätt möjliggöra att se vilka attribut som kan tänkas påverka eventuell bröstcancer.



# Material

Använder data från sklearn enligt nedan.

<https://scikit-learn.org/stable/datasets/toy_dataset.html#breast-cancer-dataset>

## 7.1.7. Breast cancer wisconsin (diagnostic) dataset

**Data Set Characteristics:**

**Number of Instances**

569

**Number of Attributes**

30 numeric, predictive attributes and the class

**Attribute Information**

* radius (mean of distances from center to points on the perimeter)
* texture (standard deviation of gray-scale values)
* perimeter
* area
* smoothness (local variation in radius lengths)
* compactness (perimeter^2 / area - 1.0)
* concavity (severity of concave portions of the contour)
* concave points (number of concave portions of the contour)
* symmetry
* fractal dimension (“coastline approximation” - 1)

The mean, standard error, and “worst” or largest (mean of the three worst/largest values) of these features were computed for each image, resulting in 30 features. For instance, field 0 is Mean Radius, field 10 is Radius SE, field 20 is Worst Radius.

* **class:**
  + WDBC-Malignant
  + WDBC-Benign

**Summary Statistics**

|  |  |  |
| --- | --- | --- |
| radius (mean): | 6.981 | 28.11 |
| texture (mean): | 9.71 | 39.28 |
| perimeter (mean): | 43.79 | 188.5 |
| area (mean): | 143.5 | 2501.0 |
| smoothness (mean): | 0.053 | 0.163 |
| compactness (mean): | 0.019 | 0.345 |
| concavity (mean): | 0.0 | 0.427 |
| concave points (mean): | 0.0 | 0.201 |
| symmetry (mean): | 0.106 | 0.304 |
| fractal dimension (mean): | 0.05 | 0.097 |
| radius (standard error): | 0.112 | 2.873 |
| texture (standard error): | 0.36 | 4.885 |
| perimeter (standard error): | 0.757 | 21.98 |
| area (standard error): | 6.802 | 542.2 |
| smoothness (standard error): | 0.002 | 0.031 |
| compactness (standard error): | 0.002 | 0.135 |
| concavity (standard error): | 0.0 | 0.396 |
| concave points (standard error): | 0.0 | 0.053 |
| symmetry (standard error): | 0.008 | 0.079 |
| fractal dimension (standard error): | 0.001 | 0.03 |
| radius (worst): | 7.93 | 36.04 |
| texture (worst): | 12.02 | 49.54 |
| perimeter (worst): | 50.41 | 251.2 |
| area (worst): | 185.2 | 4254.0 |
| smoothness (worst): | 0.071 | 0.223 |
| compactness (worst): | 0.027 | 1.058 |
| concavity (worst): | 0.0 | 1.252 |
| concave points (worst): | 0.0 | 0.291 |
| symmetry (worst): | 0.156 | 0.664 |
| fractal dimension (worst): | 0.055 | 0.208 |

**Missing Attribute Values**

None

**Class Distribution**

212 - Malignant, 357 - Benign

**Creator**

Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

**Donor**

Nick Street

**Date**

November, 1995

# Genomförande

För att visualisera 30 variabler i en två dimensionell plot skapade jag ett python projekt och plottade data med hjälp av pyplot. Men tanken blev att istället exponera PCA data via API och använda det i en React applikation likt kodexempel Sanic Spacy Client.

# Förbättringar

* 1. När det gäller serversidan så är inläsning av data till sqlite mest till för att prova på den tekniken. Så där finns det mer göra för en bättre tabellstruktur.
  2. En bit in i min implementation hittade jag följande artikel som beskriver utnyttjade av tre komponents PCA för samma dataset. Hade kunnat vara ett alternativ. <https://www.kaggle.com/jahirmorenoa/pca-to-the-breast-cancer-data-set>
  3. En hel del strul med att få plotten att fungera. Hade den först i en separat fil/funktion och använde React hook. Men då krånglade klick-funktionen.
  4. Överlag så hamnade en hel del funktionalitet i App filen och det skulle vara mer modulärt om det flyttades ut till separata funktioner.
  5. React är helt nytt för mig så det blev tydligt att en del saker blev lite omständligt. T.ex. så var jag tvungen att anropa funktionen med patientmarkörer från funktionen som skapar plotten då jag inte fick listan i state att fungera på något annat sätt.