IMPERIAL

ENSEMBLE METHODS AND UNSUPERVISED LEARNING

SURG70098 - SURGICAL DATA SCIENCE AND AI

STUART BOWYER

INTENDED LEARNING OUTCOMES

1. Understand and apply the two common ensemble learning methods (bagging and boosting)
2. Can address class imbalances
3. Can describe what is meant by unsupervised learning and why it is useful
4. Have an understanding of clustering methods and be able to apply them to data
5. Be aware of the concepts behind dimensionality reduction and when/how to apply it

MIMIC DATASET

The following code will load the datasets used in this lecture notes

In [ ]:

```
%pip install pandas_gbq

import pandas as pd
import pandas_gbq

project_id = 'mimic-project-439314'  # @param {type:"string"}

df_day1_vitalsign = pandas_gbq.read_gbq("""
  SELECT
    *,
    (dod IS NOT NULL) AND (dod <= dischtime) AS mortality,
    weight / POWER(height/100, 2) > 30 AS obese
  FROM `physionet-data.mimiciv_derived.first_day_vitalsign`
  LEFT JOIN (
    SELECT
      subject_id,
      stay_id,
      gender,
      race,
      dischtime,
      admission_age,
      dod
    FROM
      `physionet-data.mimiciv_derived.icustay_detail`
  )
  USING(subject_id, stay_id)
  LEFT JOIN (
    SELECT
      stay_id,
      AVG(weight) as weight
    FROM
      `physionet-data.mimiciv_derived.weight_durations`
    GROUP BY
      stay_id
  )
  USING(stay_id)
  LEFT JOIN (
    SELECT
      stay_id,
      CAST(AVG(height) AS FLOAT64) AS height
    FROM
      `physionet-data.mimiciv_derived.height`
    GROUP BY
      stay_id
  )
  USING(stay_id)
  WHERE heart_rate_mean IS NOT NULL
""", project_id=project_id)
```

ENSEMBLE METHODS

INTRODUCTION TO ENSEMBLE METHODS

- You probably have noticed that different models have different advantages and disadvantages
- i.e. sometimes they work well, others they do not
- Ensemble methods combine models together to improve overall performance by ...
  - Improving accuracy
  - Improving stability
  - Reducing error

How would you combine models together to optimise their group performance?

BIAS AND VARIANCE

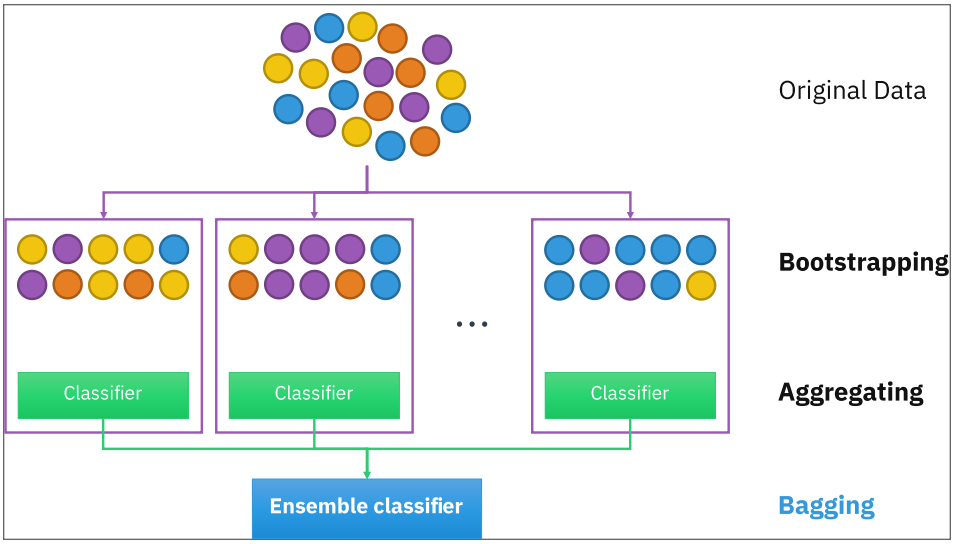
- **Bias:** when a model makes incorrect predictions due to incorrect assumptions (i.e. assuming linearity). Also called: **underfitting**
- **Variance:** when a model makes incorrect predictions due to sensitivity to small fluctuations (i.e. complex model on noisy data). Also called: **overfitting**

There is typically a **trade off** between these two aspects of a model
For example:

- Shallow decision trees are prone to high bias
- Deep decision trees are prone to high variance

BOOTSTRAP AGGREGATING (BAGGING)

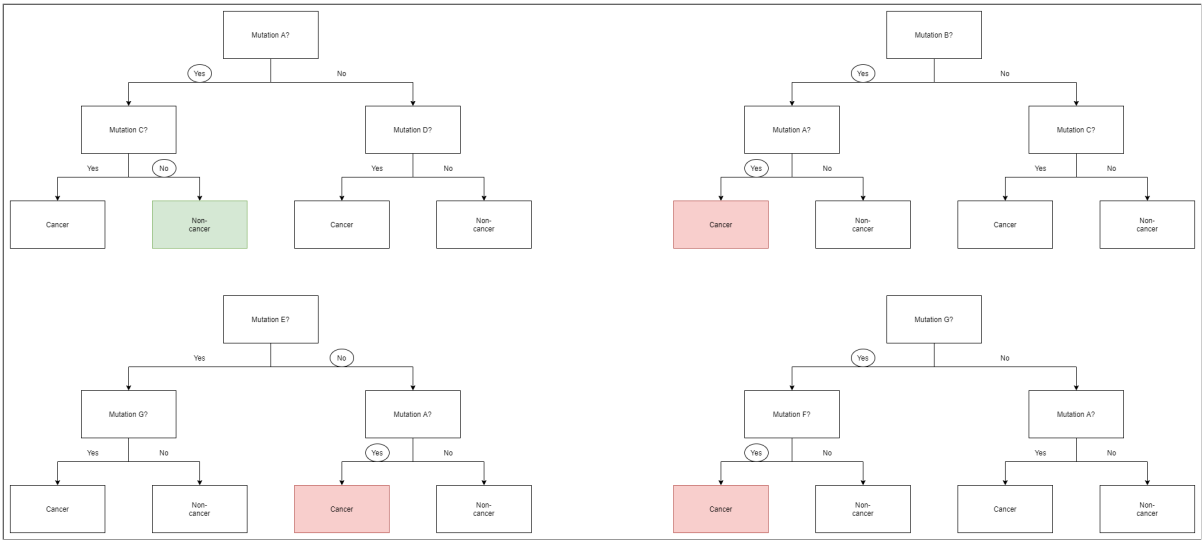
- Builds multiple parallel models independently using random (possibly overlapping) subsets of the data and combines their predictions
- Aim is to reduce variance (overfitting)
- Two components:
  - **Bootstrapping:** a random sample (with replacement) of the data is used to train several **weak learners**
  - **Aggregation:** the output of the weak learners is combined to give a single output



By Sirakorn - Own work, CC BY-SA 4.0, Link

RANDOM FORESTS

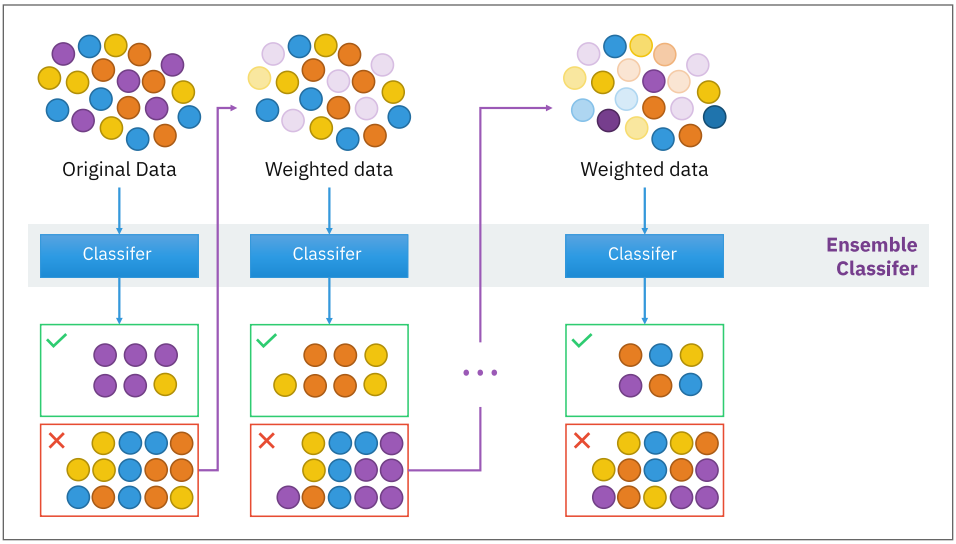
- Very commonly used with decision trees to create **random forests**
- Add the additional step of randomly selecting a subset of features for training each tree
- `from sklearn.ensemble import RandomForestClassifier`



By CollaborativeGeneticist - Own work, CC BY-SA 4.0, Link

BOOSTING

- Builds sequential models that try to correct the errors of the predecessor
- Aim is to reduce bias (underfitting) (due to weak models) by focusing models on the errors of other models



By Sirakorn - Own work, CC BY-SA 4.0, Link

GRADIENT BOOSTED TREES

- Very commonly used with decision trees to create **gradient boosted trees**
- `from sklearn.ensemble import GradientBoostingClassifier`
- Popular alternative implementation is 'XGBoost (eXtreme Gradient Boosting)' which includes regularisation

CLASS IMBALANCE

PROBLEM

- One of the common issues you will often face with adverse clinical outcomes is class imbalances
- i.e. when one of your predicted classes is much more/less common than the others
- For example, in the MIMIC dataset ...

In [100]:

```
df_day1_vitalsign.mortality.value_counts(normalize=True)
```

Out[100]:

```
mortality
False    0.883693
True     0.116307
Name: proportion, dtype: Float64
```

- This imbalance can cause training bias and poor performance

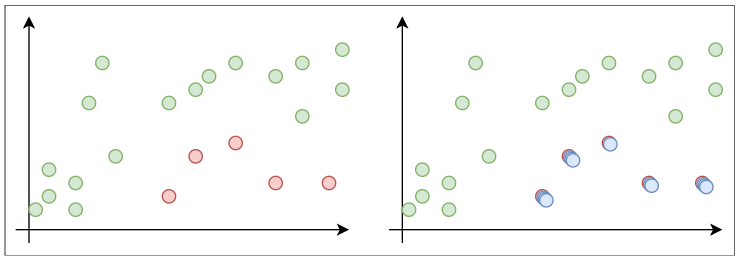
CAN YOU SUGGEST HOW TO ADDRESS THE CLASS IMBALANCE?

OVERSAMPLING

- Involves creating new observations in the minority class by ...
- **Random oversampling:** randomly duplicating entries from the minority class
- **Synthetic Minority Oversampling Technique (SMOTE):** generating new synthetic samples in the minority class by interpolating between existing observations

RANDOM OVERSAMPLING

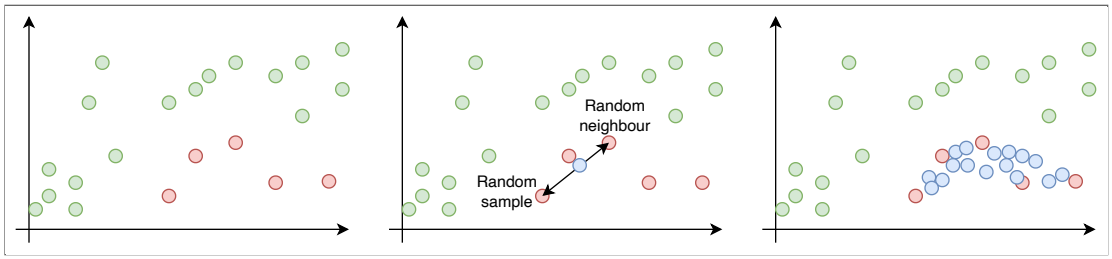
Increases size of minority dataset, but can lead to overfitting



SYNTHETIC MINORITY OVERSAMPLING TECHNIQUE (SMOTE)

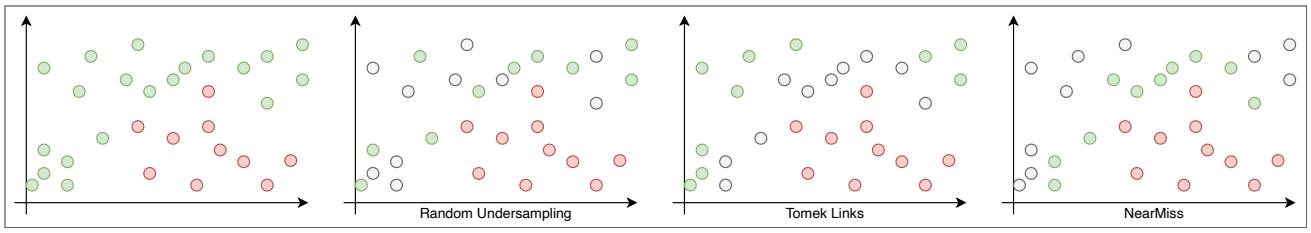
Reduces risk of overfitting, but issues remain with:

- Non-ordinal categorical data
- Noise amplification
- Dimensionality



UNDERSAMPLING

- Involves dropping observations in the majority class by ...
- **Random undersampling:** randomly removing entries from the majority class
- **Tomek links:** removes entries from the majority class that are close to the minority class (i.e. suspected noise)
- **NearMiss:** removes entries from the majority class that are far from the minority class (i.e. easy classifications)



USING `imbalanced-learn`

- Python has a package (parallelling `scikit-learn`) for addressing imbalanced datasets
  - **Oversampling methods**
  - **Undersampling methods**

In [111]:

```
from imblearn.over_sampling import RandomOverSampler

data = df_day1_vitalsign.dropna(subset=['admission_age', 'heart_rate_mean', 'sbp_mean', 'glucose_me
X = data[['admission_age', 'heart_rate_mean', 'sbp_mean', 'glucose_mean']]
Y = data['mortality']

# Create the random oversampler
ros = RandomOverSampler(random_state=1)

# Apply it to our dataset
X_res, Y_res = ros.fit_resample(X, Y)

print("Previous value counts: ", Y.value_counts().to_list())
print("Resampled value counts:", Y_res.value_counts().to_list())
```

```
Previous value counts:  [63076, 8034]
Resampled value counts: [63076, 63076]
```

EXERCISE 6.1 - SUPERVISED LEARNING CHALLENGE

Use the **Wisconsin Breast Cancer Database** to build the most high performance classifier for predicting tumour malignancy from breast mass features.
98% accuracy is possible
There are instructions on importing the dataset to Python on the above page.
I suggest starting with logistic regression on a subset of features, but you should expect to build up the model complexity and number of features.
You will probably want to use most of the techniques you have learnt in the past two/three lectures:

- Some basic EDA of this new dataset
- Ensuring the data are fully prepared
- Exploring different classification methods
- Searching for optimal parameters
- Addressing class imbalances
- Testing other performance improvements (ensemble methods)
- Using effective model validation

Before you start, what metric/s should we use?

UNSUPERVISED LEARNING

INTRODUCTION

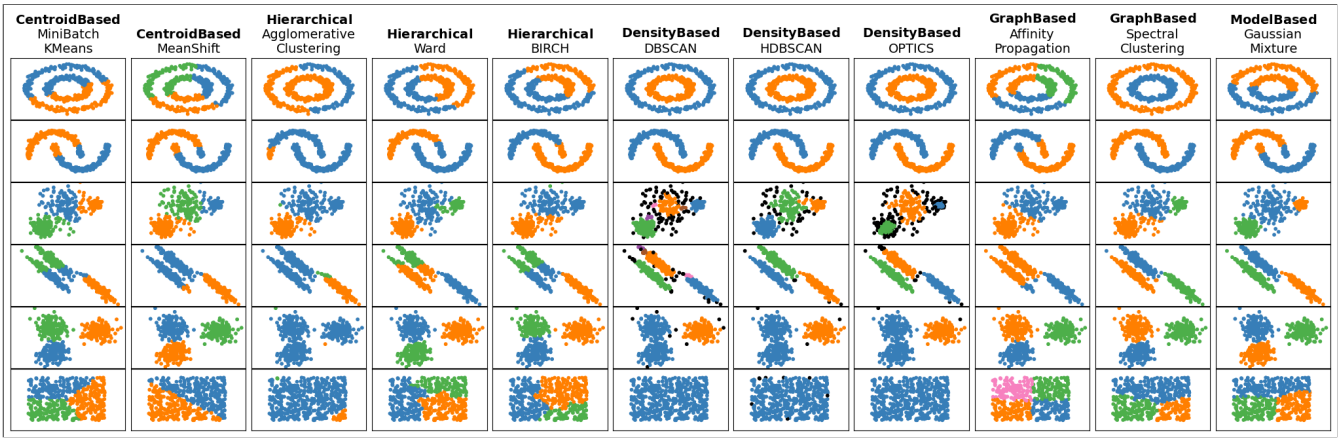
- **What:** A type of machine learning where the algorithm is trained on data without labeled outcomes
- **Why:** The goal is to uncover hidden patterns, structures, or relationships in the data **WITHOUT NEEDING LABELS**
- **Methods:**
  - **Clustering:** learning groups in the data based on common feature patterns
  - **Dimensionality Reduction:** learning simplified representations of data based on removing redundant information

CLUSTERING

- **What:** learning groupings in the data based on common feature patterns
- **Why:** reveals complex structure that is unclear from simple analyses
- **Uses:**
  - Identifying phenotypes of complex disease subtypes (e.g. in cardiac disease "young-low comorbidity burden; metabolic; cardio-renal; etc," - **https://link.springer.com/article/10.1007/s11897-023-00615-z**)
  - Segmenting structures from medical images
  - Grouping patients to predict prognosis or recovery time

CLUSTERING METHODS

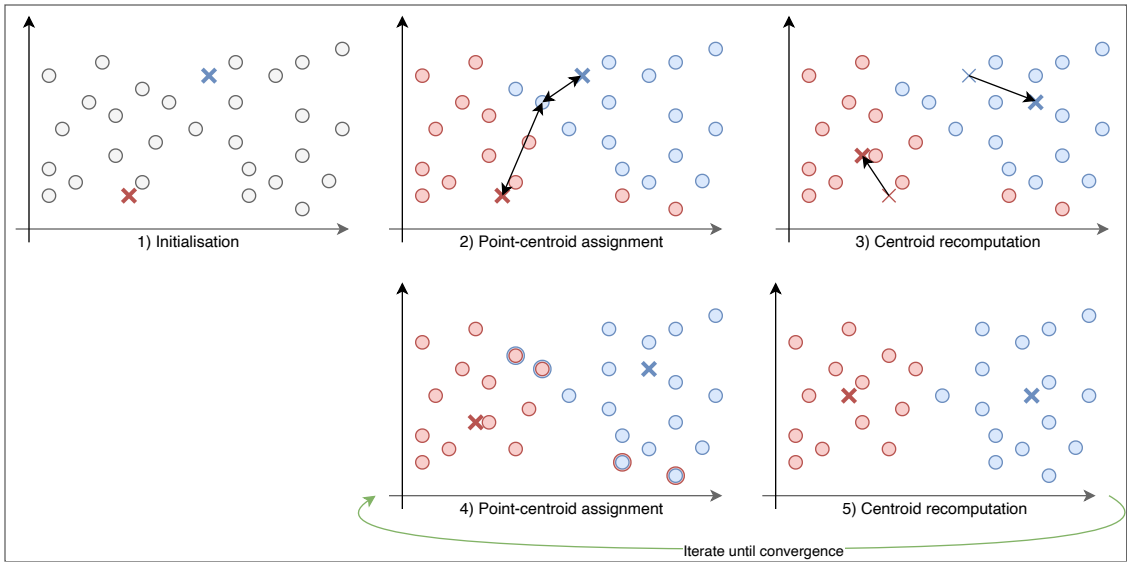
- **Centroid-Based Methods:** assign clusters based on proximity to a given centroid (e.g. K-Means)
- **Hierarchical Methods:** build clusters in a hierarchy by recursively either combining (e.g. agglomerative) or splitting (e.g. divisive) clusters
- **Density-Based Methods:** identify clusters based on contiguous areas of high-density data surrounded by low density data (e.g. DBSCAN)



Modified from **https://scikit-learn.org/1.5/auto_examples/cluster/plot_cluster_comparison.html**

K-MEANS CLUSTERING

- Centroid-based method assigns clusters based on proxmity to a given centroid
- **Centroid:** the centre of the cluster
- **Distance metric:** the numerical value defining how the 'proximity' between a data point and a centroid



Can you think of some types of cluster where this would not work well?

PROPERTIES OF K-MEANS CLUSTERING

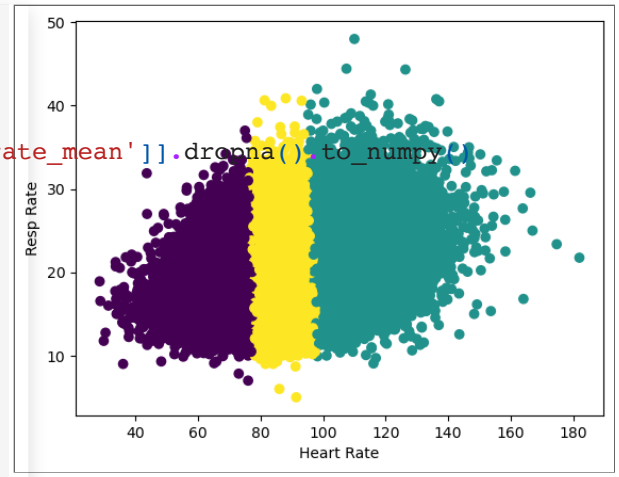
Advantages	Disadvantages
Simple and easy to implement	Requires the number of clusters ((k)) to be predefined
Computationally efficient for large datasets	Sensitive to the initial choice of centroids
Works well with spherical cluster shapes	Struggles with clusters of varying sizes or densities
Easily interpretable results	Prone to convergence to local minima
Scales well to higher-dimensional data	Affected by outliers, which can distort cluster centroids

K-MEANS IN `SCIKIT-LEARN`

What has gone wrong here?

In [244]:

```
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

# Select the two columns for analysis
X = df_day1_vitalsign[['heart_rate_mean', 'resp_rate_mean']].dropna().to_numpy()

# # Standardisation (try adding this in ...)
# scaler = StandardScaler()
# scaler.fit(X)
# X = scaler.transform(X)

# Create and fit the cluster
kmeans = KMeans(n_clusters=3, random_state=1)
kmeans.fit(X)

# Predict the cluster labels
labels = kmeans.predict(X)

# Plot the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.xlabel('Heart Rate')
plt.ylabel('Resp Rate')
plt.show()
```



DIMENSIONALITY REDUCTION

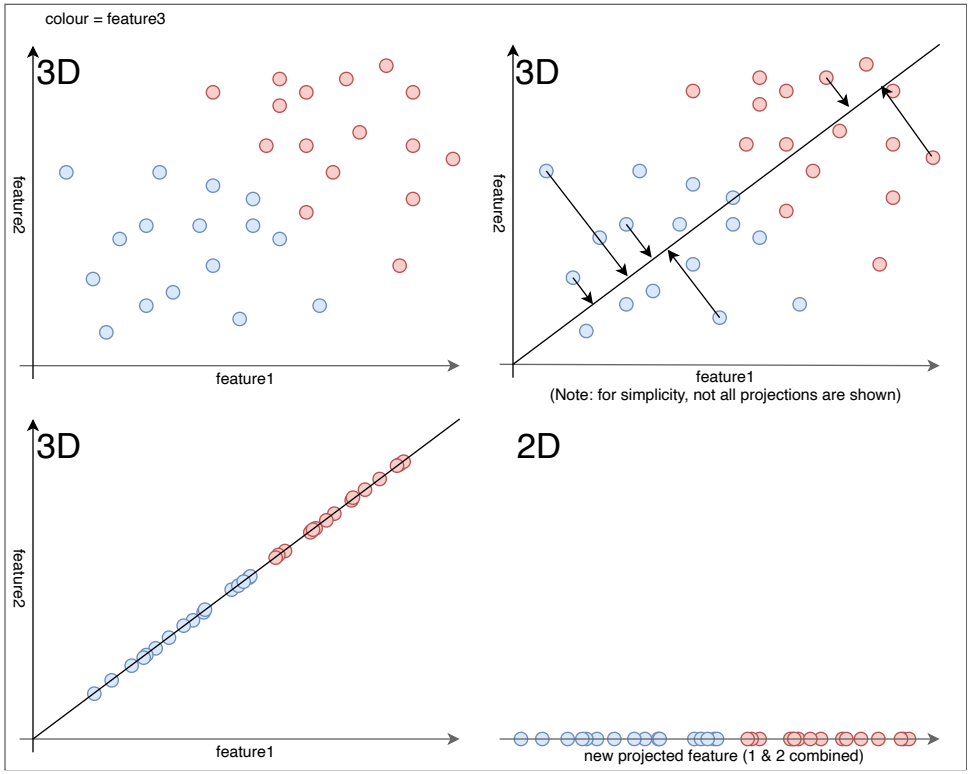
- **What:** learning simplified representations of data based on removing redundant information
- **Why:** reducing 'curse of dimensionality' and understanding feature importance and
- **Uses:**
  - Simplifying genomic data - **https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0093766**
  - Simplifying timeseries data - **https://link.springer.com/article/10.1007/s11227-021-04303-4**

FEATURE SELECTION

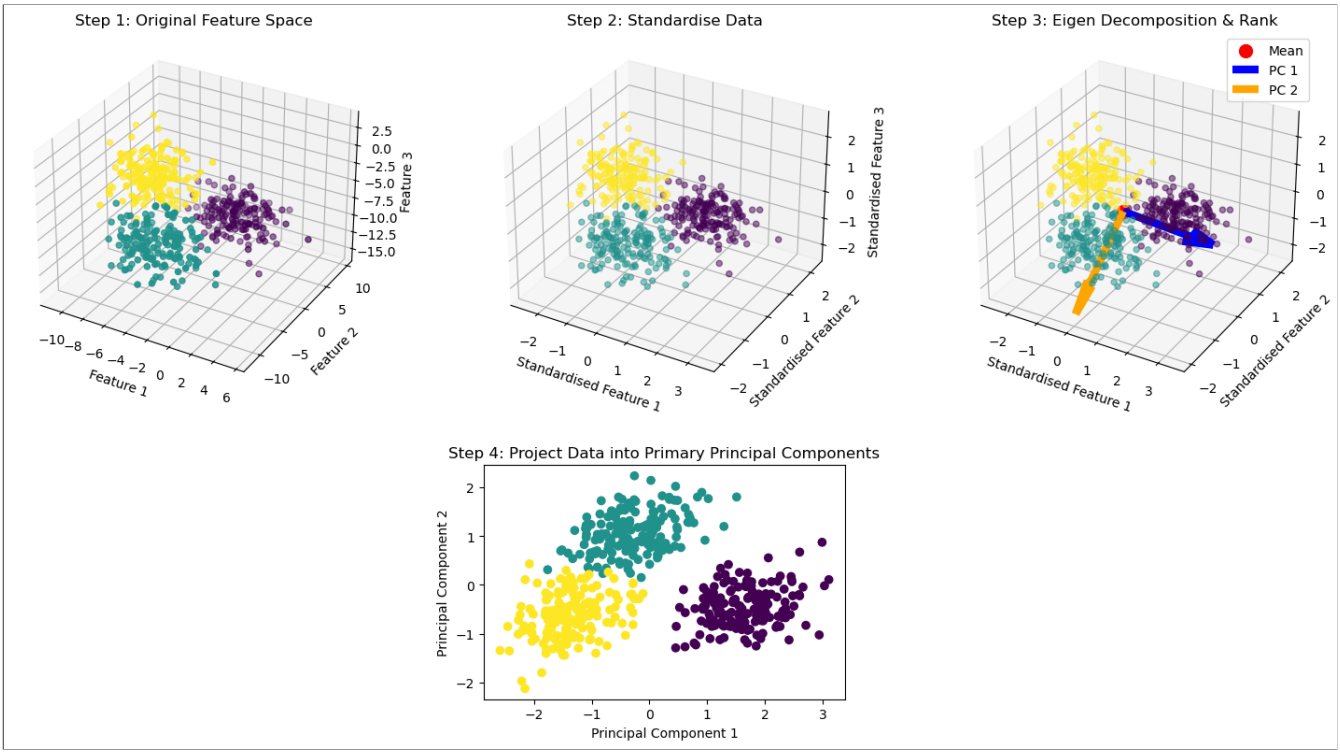
- **What:** completely removing one or more features that do not contribute 'useful information'
- As covered in previous reading for supervised learning
- **How:** various methods available that remove features with:
  - Low variance (i.e. features that do not change much)
  - High correlation (i.e. two features represent the same physical characteristic)
  - High mutual information (i.e. two features with high dependency or high shared information)

FEATURE PROJECTION

- **What:** combining multiple features together in such a way that they are still informative, but in fewer dimensions
- **How:** 'projecting' data points into a lower-dimensional space



PRINCIPAL COMPONENT ANALYSIS (PCA)



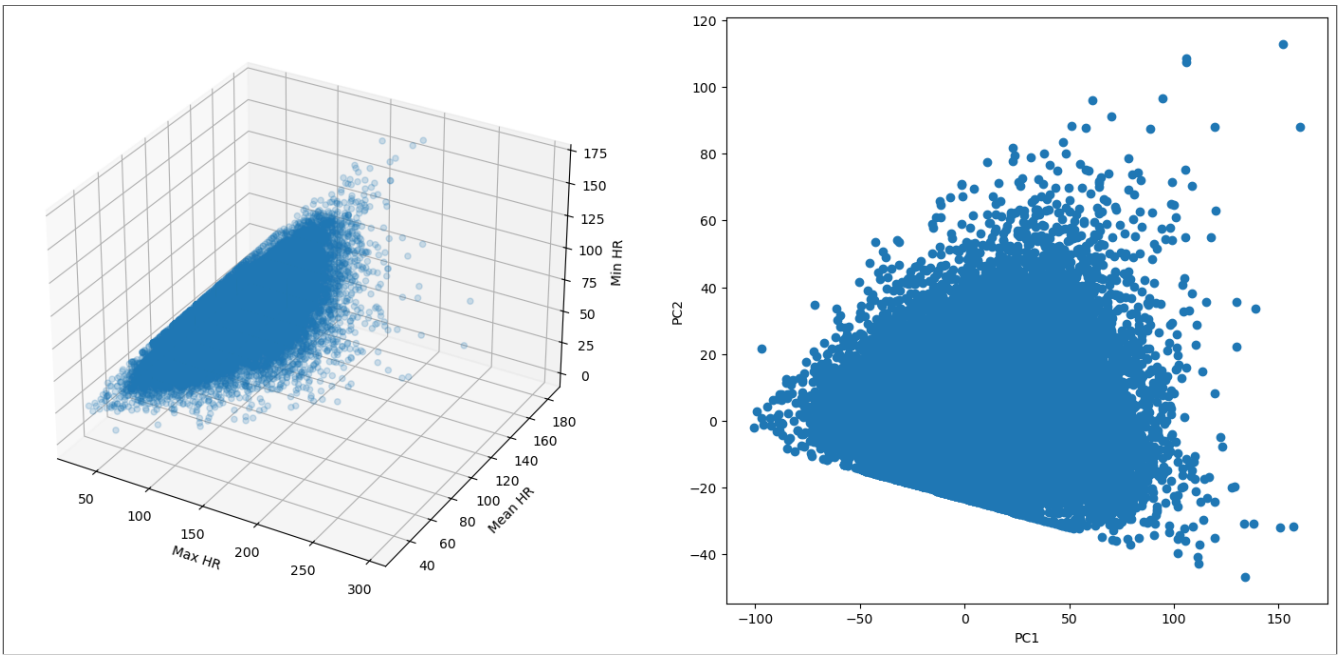
PCA IN `SCIKIT-LEARN`

In [242]:

```
%%capture out
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

# Select the two columns for analysis
X = df_day1_vitalsign[['heart_rate_max', 'heart_rate_mean', 'heart_rate_min']].dropna().to_numpy()

# Create and fit the PCA
pca = PCA(n_components=2)
pca.fit(X)

# Print the Explained Variance Ratio
print(f"Explained variance ratio: {pca.explained_variance_ratio_}")

# Transform the data
X_trans = pca.transform(X)

# Plot the Full Heart Rate Data
fig = plt.figure(figsize=(18, 8))
ax1 = fig.add_subplot(121, projection='3d')
ax1.scatter(X[:, 0], X[:, 1], X[:, 2], alpha=0.2)
ax1.set_xlabel("Max HR")
ax1.set_ylabel("Mean HR")
ax1.set_zlabel("Min HR")

# Plot the Dimension Reduced Data
ax2 = fig.add_subplot(122)
ax2.scatter(X_trans[:, 0], X_trans[:, 1])
ax2.set_xlabel('PC1')
ax2.set_ylabel('PC2')
plt.show()
```

OUTPUT

```
Explained variance ratio: [0.84141058 0.14222821]
```



EXERCISE 6.2 - UNSUPERVISED CLUSTERING OF THE WISCONSIN BREAST CANCER DATABASE

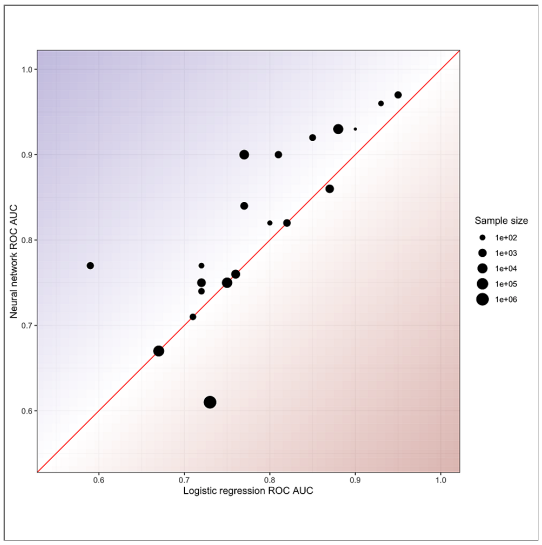
Use K-means clustering to cluster **only the features** from the Wisconsin Breast Cancer Database, that we used earlier for supervised learning.
First try with 2, then increase the number of clusters.
Once you have generated a cluster do some very basic EDA to explore how your clusters associate with the tumour type (output variable).
What do you notice about these clusters?
What does this suggest about the tumour types?

EXERCISE 6.3 - DIMENSIONALITY REDUCTION OF THE WISCONSIN BREAST CANCER DATABASE

Use the PCA dimensionality reduction method to see how many dimensions you can reduce from the Wisconsin Breast Cancer Database before the performance of your classifier from Exercise 6.1 loses a lot of performance.
Start by reducing the dimensionality by 1 and compute the classifier's performance, then keep reducing the dimensionality.
You should then produce a plot of 'number of dimensions' against 'classifier performance'.

WRAP UP

- You now have lots of tools to address supervised and unsupervised machine learning problems; however, experience and reading up on the nuances of each method is the difference between using them well
- Data preparation is critically important to model performance
- Consider the bias-variance trade offs
- REMEMBER... if your model does not have the key features of the system/disease/procedure/etc. you are trying to predict, no matter how complex you make your model or how many samples you collect, it will never improve its performance



Issitt R W, Cortina-Borja M, Bryant W, et al. (February 21, 2022) Classification Performance of Neural Networks Versus Logistic Regression Models: Evidence From Healthcare Practice. Cureus 14(2): e22443. doi:10.7759/cureus.2244

SELF STUDY

NEW MATERIAL

- Detail on Gradient Boosting / XGboost - **https://xgboost.readthedocs.io/en/stable/tutorials/model.html**
- Silhouette scoring of cluster separation - **https://scikit-learn.org/1.5/auto_examples/cluster/plot_kmeans_silhouette_analysis.html**
- t-SNE (t-distributed Stochastic Neighbor Embedding) tutorial - **https://www.datacamp.com/tutorial/introduction-t-sne**

CONSOLIDATION READING

- Original SMOTE Oversampling Method - **https://arxiv.org/pdf/1106.1813**
- Boosting and Bagging Paper - **https://www.d.umn.edu/~rmaclin/cs5751/notes/opitz-jair99.pdf**
- Survey article on using clustering for discovering clinical phenotypes in cardiac disease - **https://link.springer.com/article/10.1007/s11897-023-00615-z**
- `scikit-learn`'s high-level overview of clustering methods - **https://scikit-learn.org/1.5/modules/clustering.html**