Interpretable Machine Learning Model to Predict and Influence Mortality of Patients with Heart Failure Warded in Intensive Care Unit

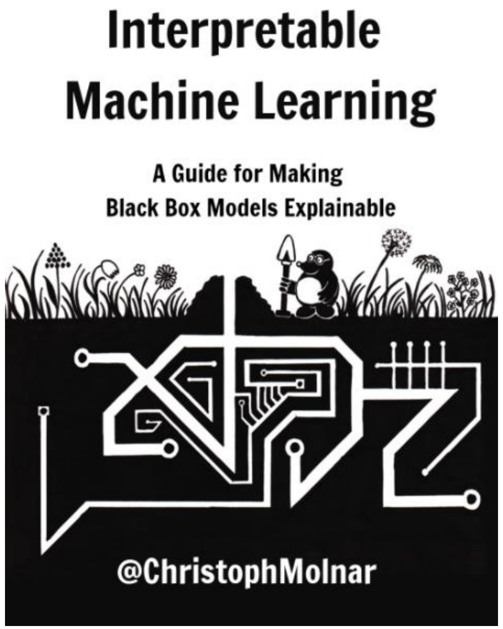
WONG CHI KEONG JOHN
3 JAN 2022

GENERAL ASSEMBLY
DATA SCIENCE IMMERSIVE FLEX 2
CAPSTONE PROJECT

Data Science Process

- **Objective**
- Collect
- Explore
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

- Develop interpretable machine learning model to predict and influence mortality of patients with heart failure warded in intensive care unit
- Interpretable
  - *“Interpretability is the degree to which a human can understand the cause of a decision.”*
  - *“Interpretability is the degree to which a human can consistently predict the model’s result”*



Data Science Process

- Objective
- **Collect**
- Explore
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

- Data from Kaggle
  - https://www.kaggle.com/saurabhshahane/in-hospital-mortality-prediction
- Paper from journal
  - https://bmjopen.bmj.com/content/11/7/e044779

BMJ Open

Prediction model of in-hospital mortality in intensive care unit patients with heart failure: machine learning-based, retrospective analysis of the MIMIC-III database

Fuhai Li,[1,2] Hui Xin,[1] Jidong Zhang,[1] Mingqiang Fu,[2] Jingmin Zhou ,[2] Zhexun Lian [1]

Data Science Process

- Objective
- Collect
- **Explore**
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
  - 51 variables × 1,177 cases
- Variable type, category count, encoding
- Missing
- Duplicate
- Imbalance
- Multicollinearity

Data Science Process

- Objective
- Collect
- **Explore**
- **Engineer feature**
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
  - 11 features $\Rightarrow$ Categorical, binary, encoded in numeric
  - 38 features $\Rightarrow$ Continuous
  - Target $\Rightarrow$ Categorical, binary, encoded in numeric
  - ID feature $\Rightarrow$ Drop
- Missing
- Duplicate
- Imbalance
- Multicollinearity

No need one hot encoding

Data Science Process

- Objective
- Collect
- **Explore**
- **Engineer feature**
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
- Missing
  - 8 features have missing values from 12.2% to 25.0%
  - 11 features have missing values from 0.1% to 3.1%
  - Target (“outcome”) has 1 missing value
- Duplicate
- Imbalance
- Multicollinearity

Drop because too much to impute without affecting original probability distribution

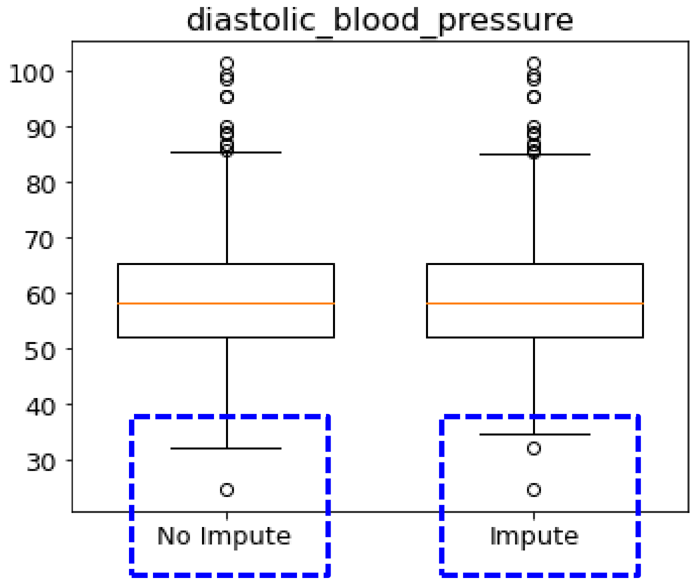
Data Science Process

- Objective
- Collect
- **Explore**
- **Engineer feature**
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
- Missing
  - 8 features have missing values from 12.2% to 25.0%
  - 11 features have missing values from 0.1% to 3.1%
  - Target (“outcome”) has 1 missing value
- Duplicate
- Imbalance
- Multicollinearity

Impute using KNNImputer with neighbouring samples = 5

Check original probability distribution not significantly affected (change in outliers, percentile and whiskers values)



Data Science Process

- Objective
- Collect
- **Explore**
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
- Missing
  - 8 features have missing values from 12.2% to 25.0%
  - 11 features have missing values from 0.1% to 3.1%
  - Target ("outcome") has 1 missing value
- Duplicate
- Imbalance
- Multicollinearity

Drop missing case before train test split

Data Science Process

- Objective
- Collect
- **Explore**
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
- Missing
- Duplicate
  - No duplicate cases
- Imbalance
- Multicollinearity

Data Science Process

- Objective
- Collect
- **Explore**
- **Engineer feature**
- Build model
- Conclude on objective
- Deployment workflow

- Data frame size
- Variable type, category count, encoding
- Missing
- Duplicate
- Imbalance
  - Die (“1”) at 13.5%
  - Live (“0”) at 86.5%
- Multicollinearity

Need to oversample minority category using SMOTE

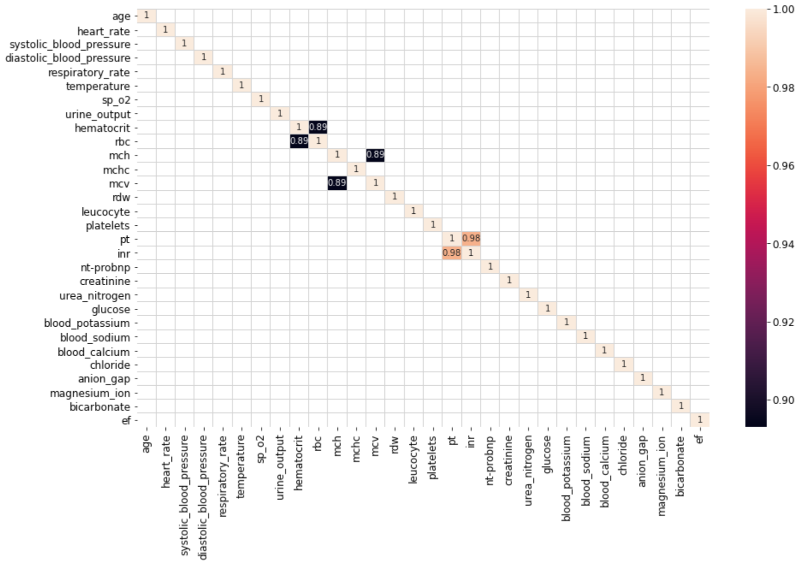
Data Science Process

- Objective
- Collect
- **Explore**
- Engineer feature
- Build model
- Conclude on objective
- Deployment workflow

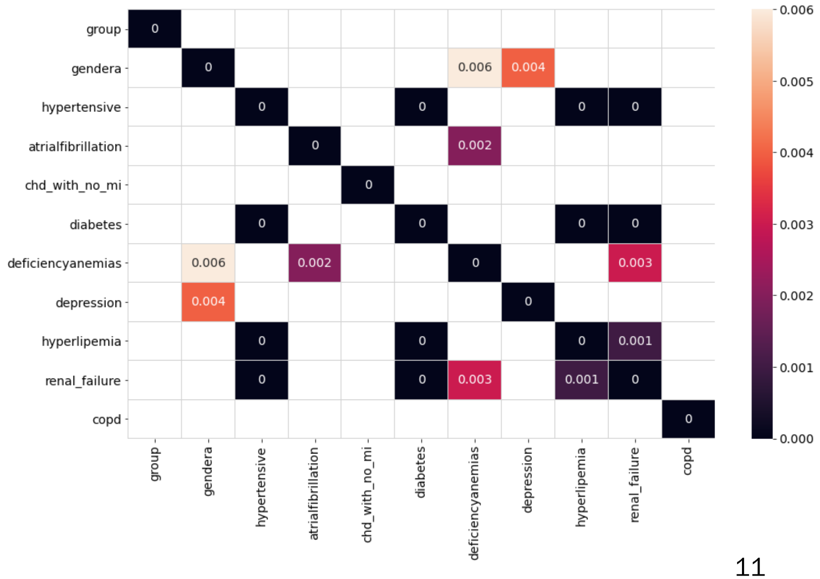
- Data frame size
- Variable type, category count, encoding
- Missing
- Duplicate
- Imbalance
- Multicollinearity

Need to regularise with L1 (LASSO) as many features are strongly correlated with each other

Pearson (continuous, more than $R^2 \geq 0.80$)



Chi Square (categorical, less than $p < 0.01$)



Data Science Process

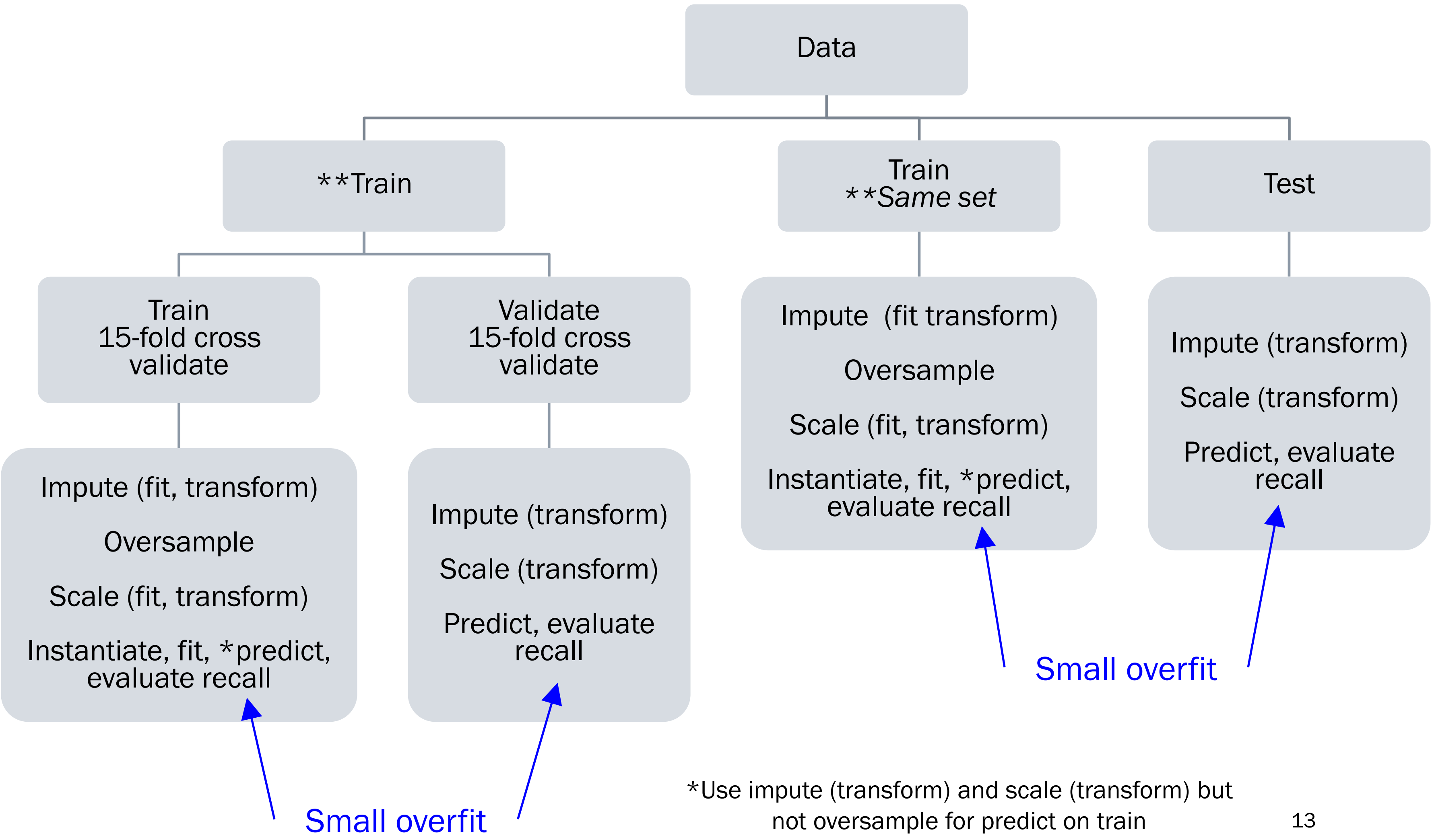
- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
- Conclude on objective
- Deployment workflow

- Model requirement
  - Want high recall $\Rightarrow$ TP $\div$ (TP + FN)
    - FN $\Rightarrow$ Cannot save patient who will actually die
  - Balance with precision $\Rightarrow$ TP $\div$ (TP + FP)
    - FP $\Rightarrow$ Unnecessary extra resources spent on patient who will actually live
  - Few important features to facilitate interventions to change mortality
  - Smallest overfit

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
- Conclude on objective
- Deployment workflow

- Model workflow
  - Mechanics of Pipeline and GridSearchCV to do manually for model in statsmodels.api



Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
- Conclude on objective
- Deployment workflow

- Model performance

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Explore different sampling strategy	• StandardScaler	• Explore different hyper-parameters for different models	• Explore different decision boundary

Prediction
Output = Probability

Prediction
Output = Category

Probability >=
Decision Boundary
Category = 1 (Die)

Probability <
Decision Boundary
Category = 0 (Live)

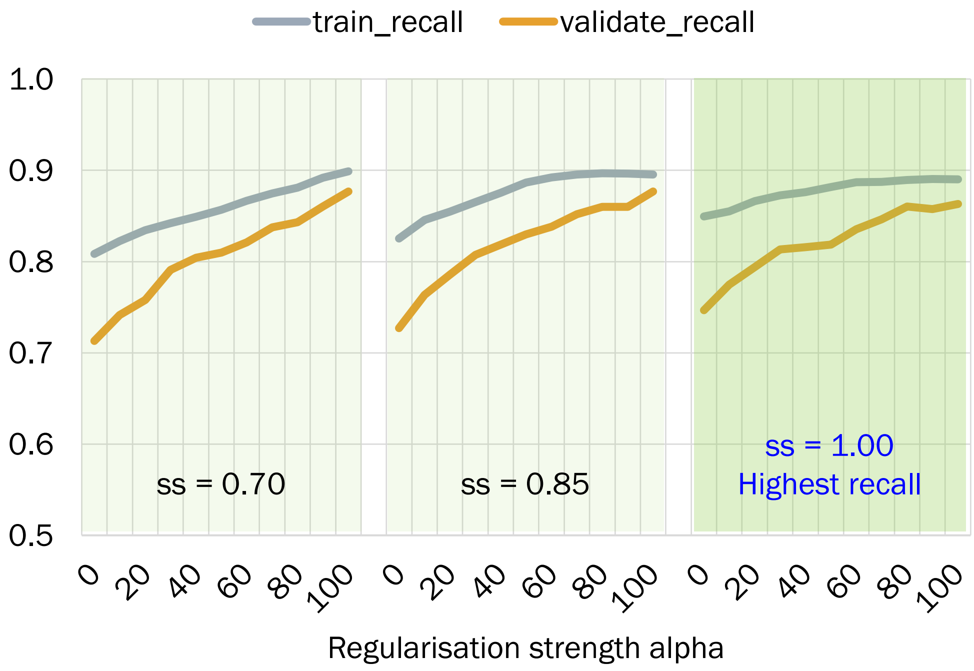
Same case
Different probability for different oversample and model
Different category for same decision boundary
Hence will explore together

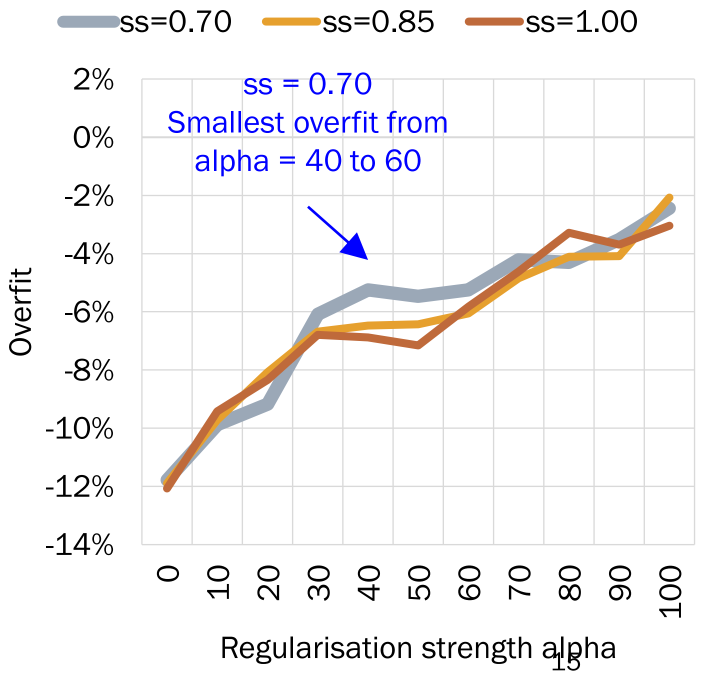
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - **Logistic Regression**
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Logistic Regression with Regularisation (statsmodels.api)

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 0.70, 0.85, 1.00	• StandardScaler	• Regularisation strength • L1 Alpha = 0 to 100	• Probability > 0.50, 0.35, 0.20 ⇒ Category = 1 (Die)



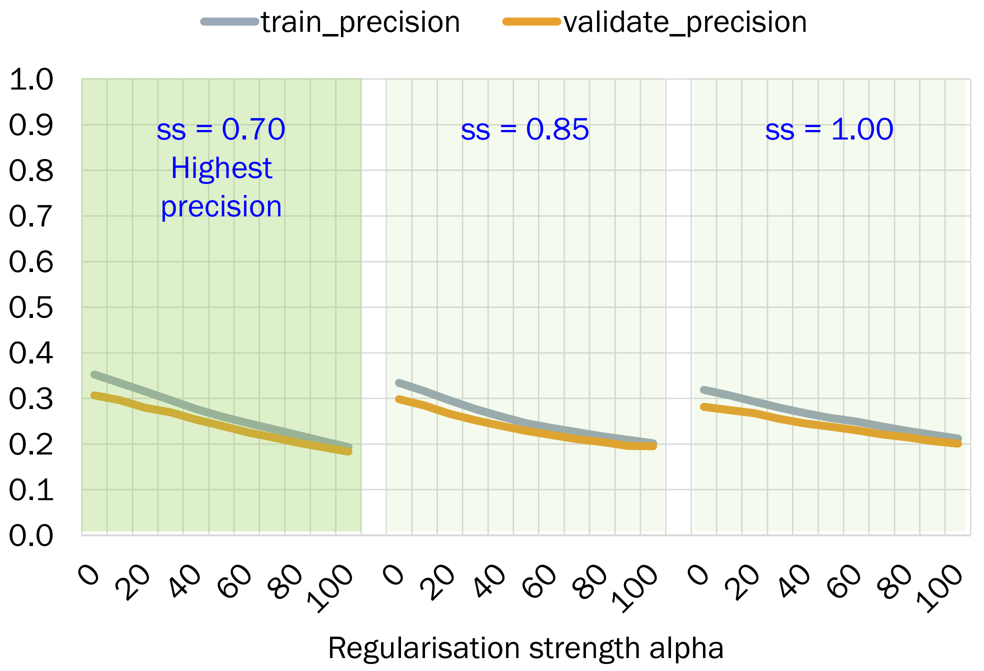


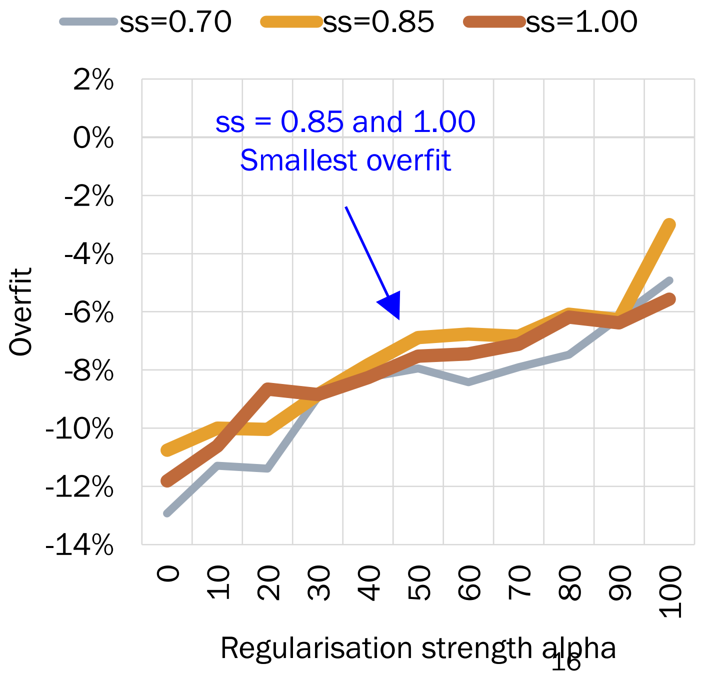
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - **Logistic Regression**
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Logistic Regression with Regularisation (statsmodels.api)

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 0.70, 0.85, 1.00	• StandardScaler	• Regularisation strength • L1 Alpha = 0 to 100	• Probability > 0.50, 0.35, 0.20 ⇒ Category = 1 (Die)



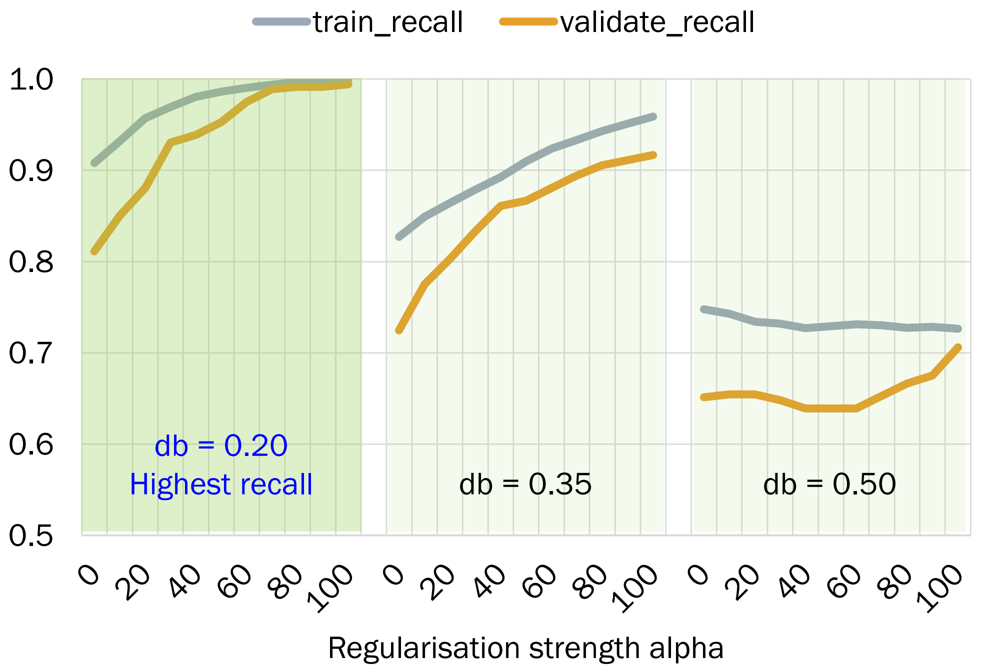


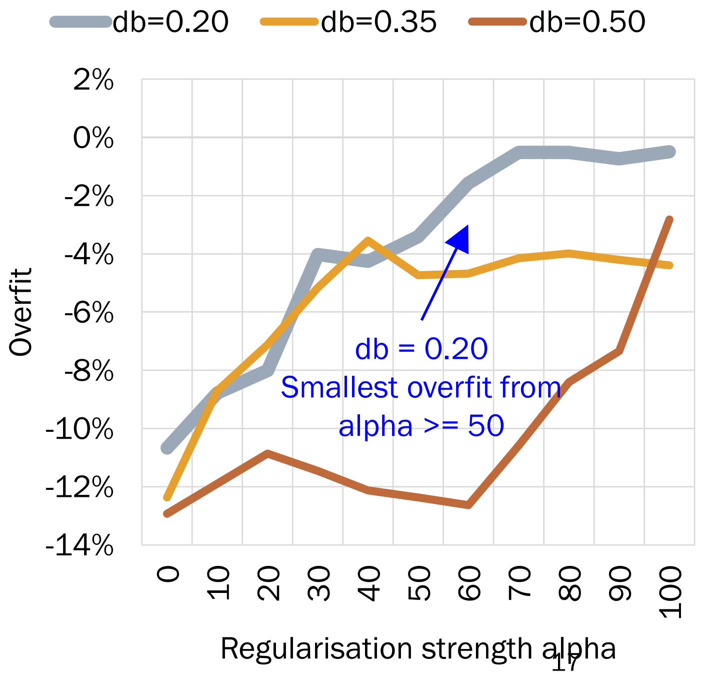
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - **Logistic Regression**
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Logistic Regression with Regularisation (statsmodels.api)

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 0.70, 0.85, 1.00	• StandardScaler	• Regularisation strength • L1 Alpha = 0 to 100	• Probability > 0.50, 0.35, 0.20 ⇒ Category = 1 (Die)





Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - **Logistic Regression**
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Logistic Regression with Regularisation (statsmodels.api)

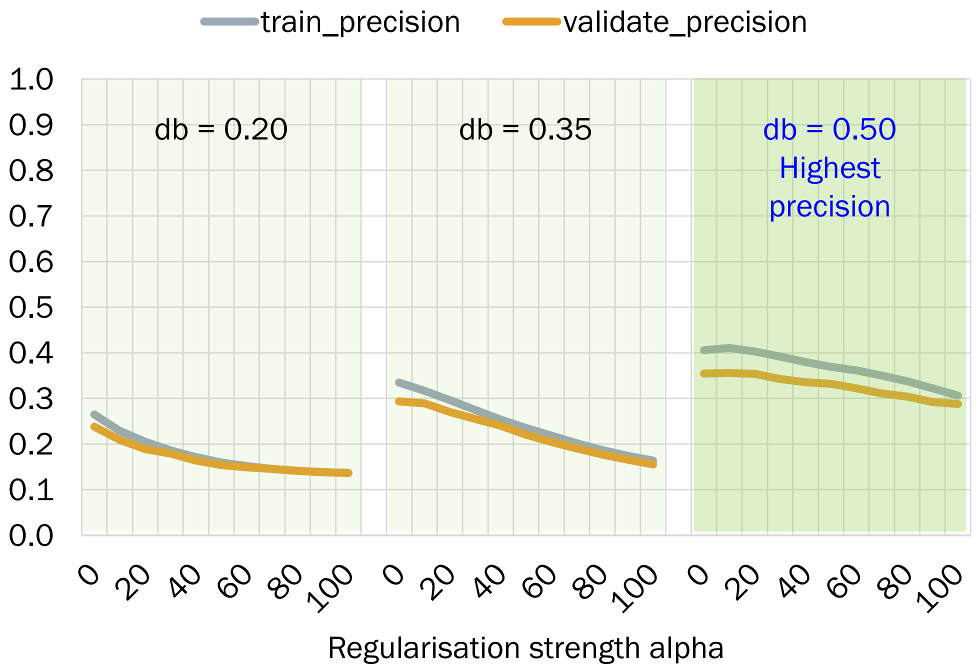
Impute
- KNNImputer
- Neighbouring samples = 5

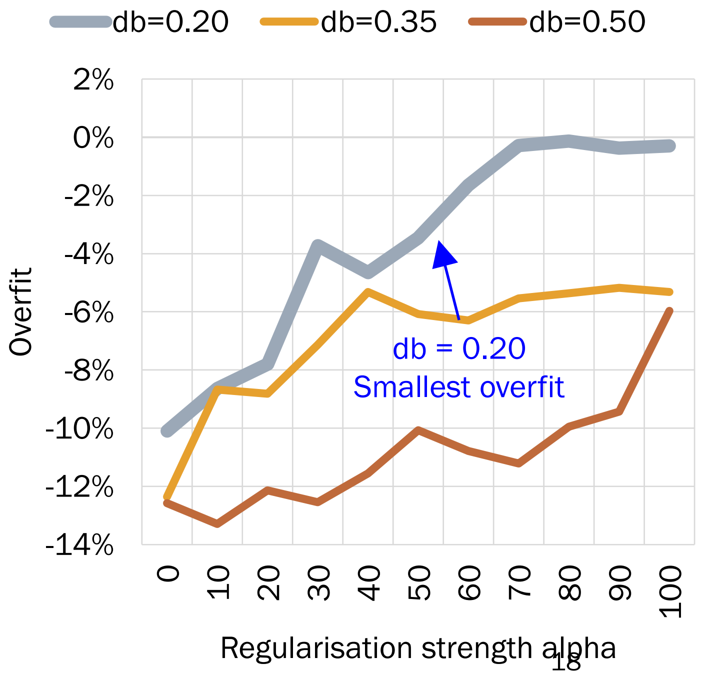
Oversample
- SMOTE
- Sampling strategy = 0.70, 0.85, 1.00

Scale
- StandardScaler

Model
- Regularisation strength
- L1 Alpha = 0 to 100

Decision Boundary
- Probability > 0.50, 0.35, 0.20 ⇒ Category = 1 (Die)



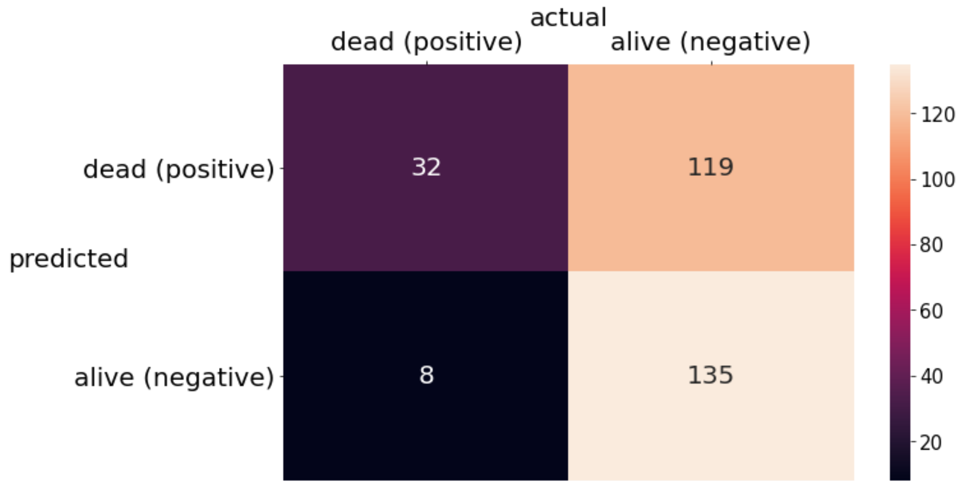


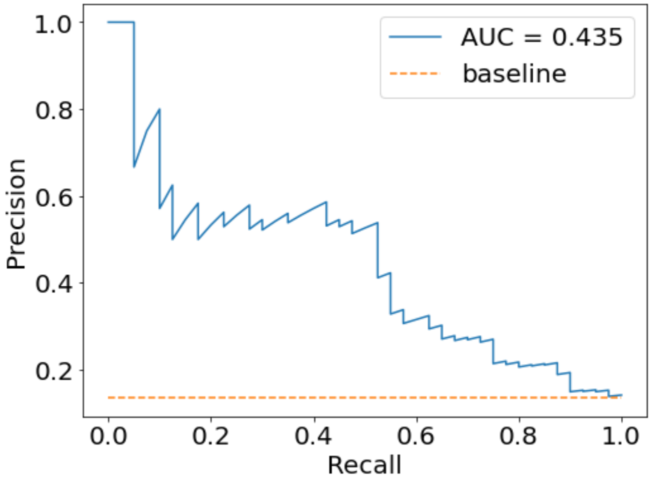
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - **Logistic Regression**
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Logistic Regression with Regularisation (statsmodels.api)

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 0.85 (smallest overfit)	• StandardScaler	• Regularisation strength • L1 Alpha = 50 simpler model but not too simple	• Probability > 0.35 ⇒ Category = 1 (Die) (balance overfit, recall, precision)





Recall (overfit)
CV Train = 0.920
CV Validate = 0.875 (-4.85%)

Recall (overfit)
Train = 0.915
Test = 0.800 (-12.7%)

Precision (overfit)
Train = 0.226
Test = 0.212 (-6.09%)

Data Science Process

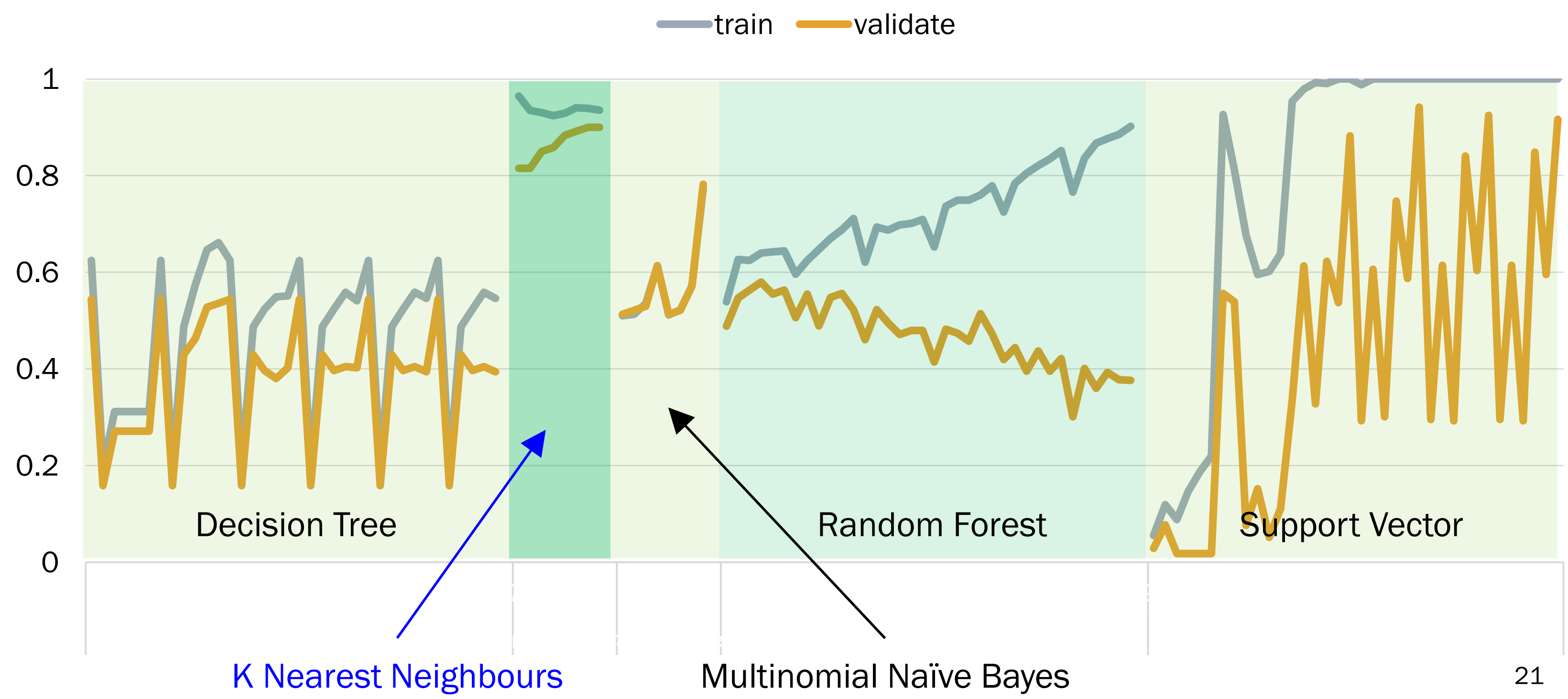
- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Pipeline (train=fit_transform, validate=transform)
  - KNNImpute
  - SMOTE
  - StandardScaler
  - Model
    - Decision Tree
    - Random Forest
    - Multinomial Naïve Bayes
    - K Nearest Neighbours
    - Support Vector Machine
- Hyperparameter search settings
- GridSearchCV (return train score, scoring = recall, 15-fold)
- Fit
- CV results
  - Train recall
  - Validate recall

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

- Model
  - Decision Tree
  - Random Forest
  - Multinomial Naïve Bayes
  - K Nearest Neighbours (best balanced performance in recall and overfit)
  - Support Vector Machine



Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - **K Nearest Neighbours**
  - Neural Network
- Conclude on objective
- Deployment workflow

- K Nearest Neighbours

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 0.70, 0.85, 1.00	• StandardScaler	• Nearest neighbours = 100, 150, 200 • Weight = Uniform, Distance • Algorithm = Auto, Brute • P = 1, 2	• Default in K Nearest Neighbours

Over	Train	Valid	Over
0.70	0.863	0.698	-19.14%
0.85	0.915	0.792	-13.42%
1.00	0.941	0.848	-9.98%

NN	Train	Valid	Over
100	0.908	0.774	-14.76%
150	0.907	0.780	-13.99%
200	0.905	0.784	-13.37%

p	Train	Valid	Over
p=1	0.865	0.688	-20.49%
p=2	0.948	0.871	-8.17%

Weight	Train	Valid	Over
Dist	1.000	0.781	-21.87%
Unif	0.813	0.777	-4.42%

Algo	Train	Valid	Over
auto	0.907	0.779	-14.04%
brute	0.907	0.779	-14.04%

Parameter and hyperparameters with smallest overfit indicated in blue

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - **K Nearest Neighbours**
  - Neural Network
- Conclude on objective
- Deployment workflow

- K Nearest Neighbours

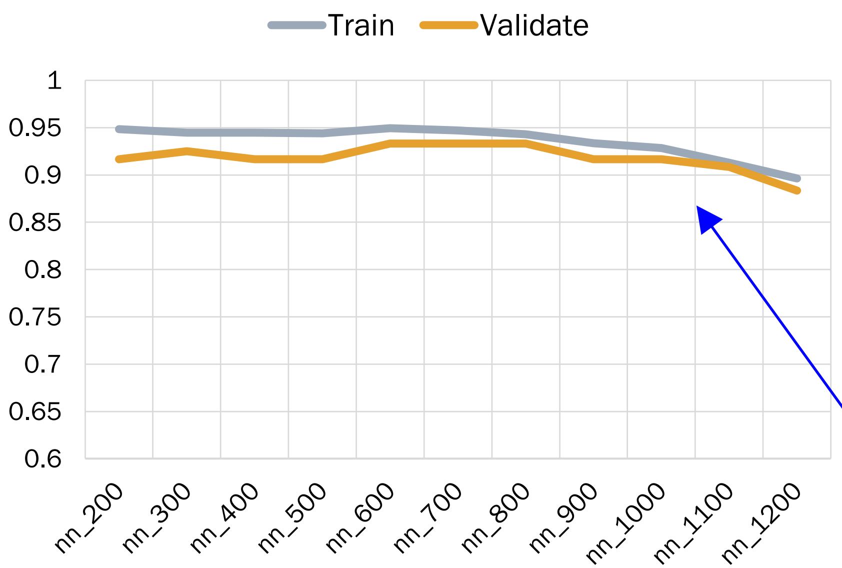
Impute
- KNNImputer
- Neighbouring samples = 5

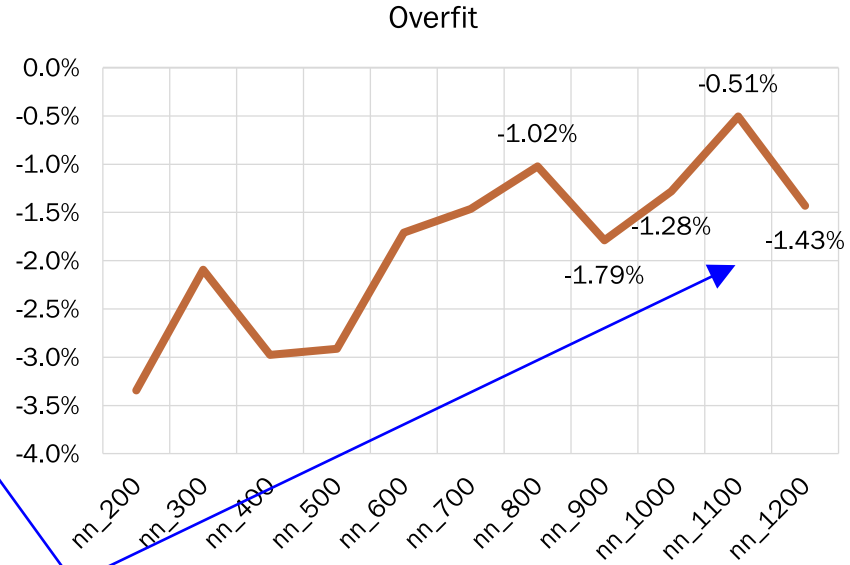
Oversample
- SMOTE
- Sampling strategy = 1.00 (smallest overfit)

Scale
- StandardScaler

Model
- Nearest neighbours = 200 to 1200 (to refine further)
- Weight = Uniform (smallest overfit)
- Algorithm = Brute
- P = 2 (smallest overfit)

Decision Boundary
- Default in K Nearest Neighbours





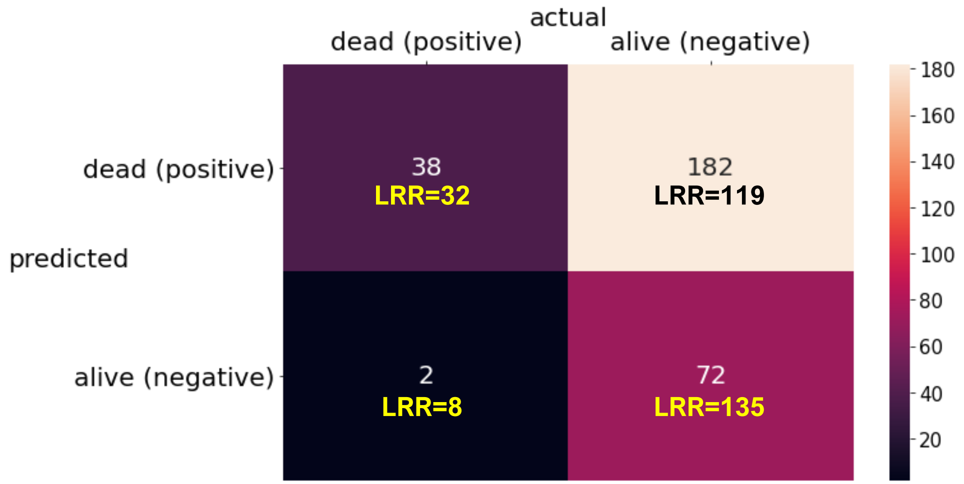
More nearest neighbours, lower performance and smaller overfit

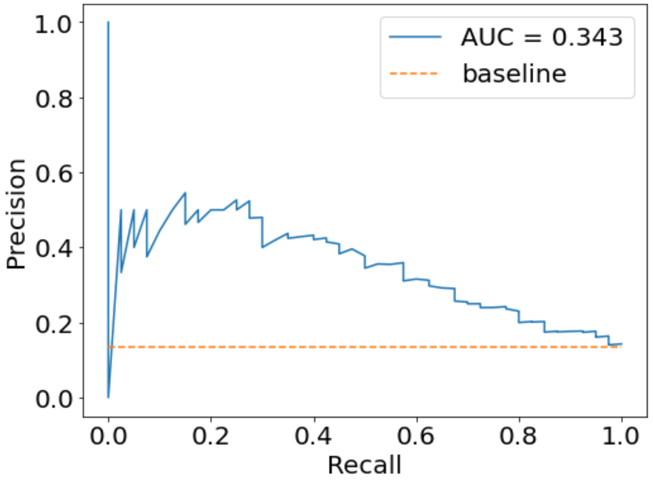
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - **K Nearest Neighbours**
  - Neural Network
- Conclude on objective
- Deployment workflow

- K Nearest Neighbours

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 1.00 (smallest overfit)	• StandardScaler	• Nearest neighbours = 1000 (smallest overfit) • Weight = Uniform (smallest overfit) • Algorithm = Brute • P = 2 (smallest overfit)	• Default in K Nearest Neighbours





Recall (overfit)
CV Train = 0.929
CV Validate = 0.917 (-1.29%)

Recall (overfit)
Train = 0.933
Test = 0.950 (1.85%)

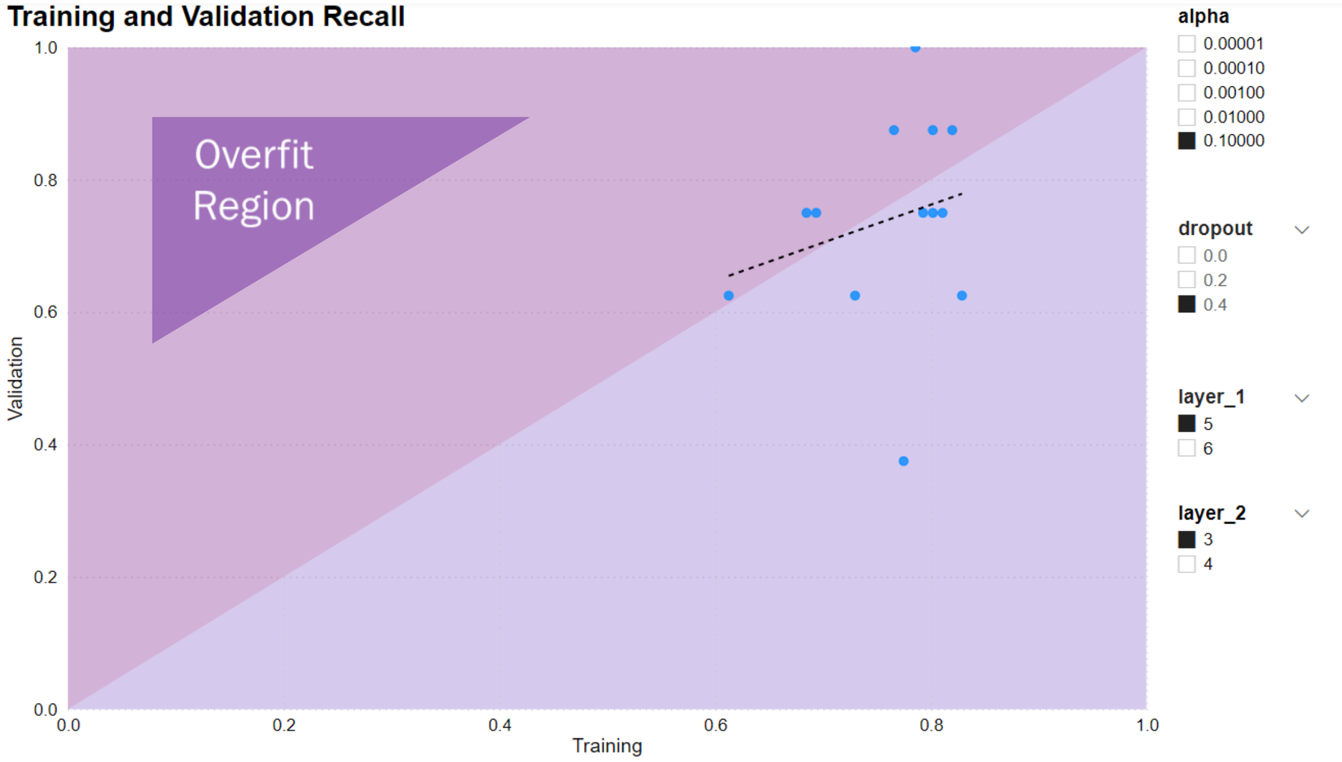
Precision (overfit)
Train = 0.171
Test = 0.173 (1.15%)

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - K Nearest Neighbours
  - **Neural Network**
- Conclude on objective
- Deployment workflow

- Neural Network with Regularisation, Dropout Rate, Early Stopping

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 1.00	• StandardScaler	• Layer 1 = 5, 6 • Layer 2 = 3, 4 • Dropout = 0.0, 0.2, 0.4 • L1 Alpha = 0.1, 0.01, 0.001, 0.001, 0.0001	• 0.4



Visualisation with Microsoft Power BI

Good balance of
* slope = 1
* small overfit

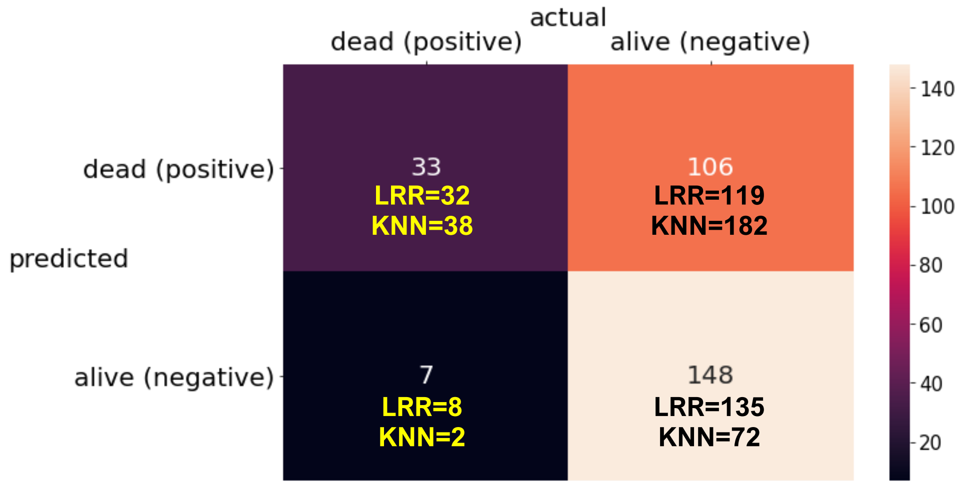
Best hyperparameters with
* layer1 = 5 neurons
* layer2 = 3 neurons
* dropout = 0.4
* L1 alpha = 0.1

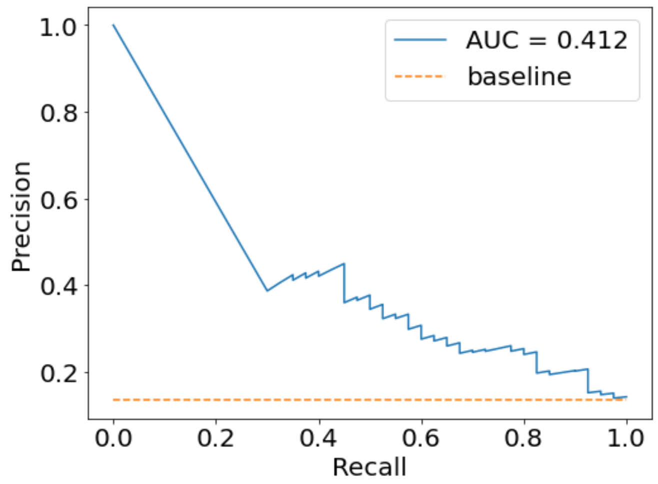
Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- **Build model**
  - Logistic Regression
  - K Nearest Neighbours
  - **Neural Network**
- Conclude on objective
- Deployment workflow

- Neural Network with Regularisation, Dropout Rate, Early Stopping

Impute	Oversample	Scale	Model	Decision Boundary
• KNNImputer • Neighbouring samples = 5	• SMOTE • Sampling strategy = 1.00	• StandardScaler	• Layer 1 = 5 • Layer 2 = 3 • Dropout = 0.4 • Alpha = 0.1	• 0.4





Recall (overfit)
CV Train = 0.769
CV Validate = 0.742 (-3.54%)

Recall (overfit)
Train = 0.924
Test = 0.825 (-10.8%)

Precision (overfit)
Train = 0.264
Test = 0.237 (-10.0%)

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- **Conclude on objective**
- Deployment workflow

- Develop interpretable machine learning model to predict and influence mortality of patients with heart failure warded in intensive care unit
- **Logistic Regression using statistical significance**

feature	std_coef	z	p
renal_failure	-0.3382	-4.851	0
bicarbonate	-0.2817	-3.114	0.002
blood_calcium	-0.2525	-3.602	0.000
urine_output	-0.2168	-3.133	0.002
deficiencyanemias	-0.1804	-2.871	0.004
sp_o2	-0.1459	-2.245	0.025
heart_rate	0.1387	2.031	0.042
leucocyte	0.1719	2.298	0.022
atrialfibrillation	0.1725	2.785	0.005
urea_nitrogen	0.3728	4.485	0.000

renal_failure:
- Categorical, Negative
- Presence will reduce odds of dying compared to absence

bicarbonate:
- Numeric, Negative
- Higher value will reduce odds of dying

heart_rate:
- Numeric, Positive
- Higher value will increase odds of dying

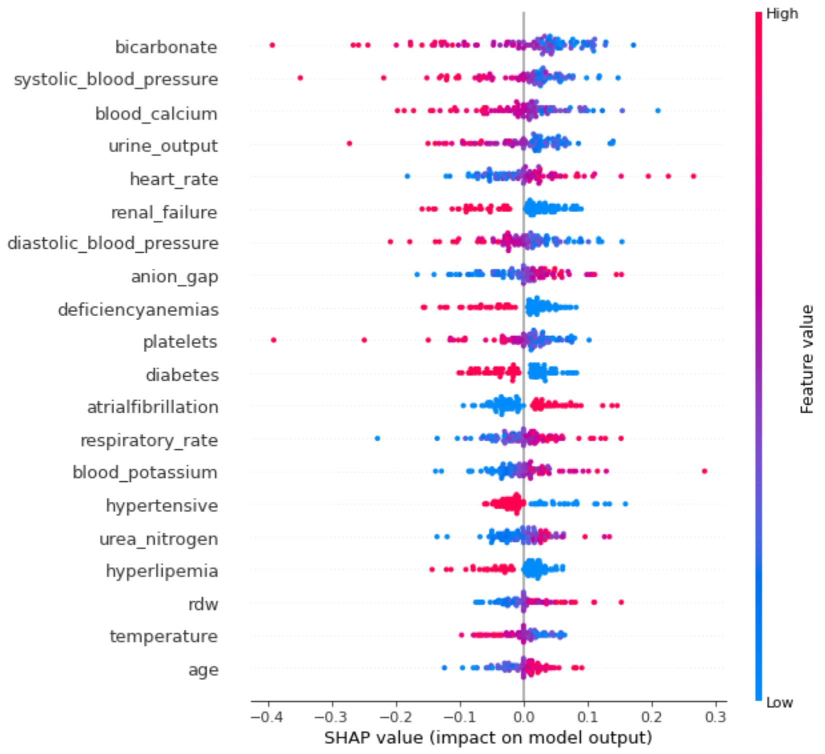
atrialfibrillation:
- Categorical, Positive
- Presence will increase odds of dying compared to absence

- Interpretable
  - Causes of death are understood

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- **Conclude on objective**
- Deployment workflow

- Develop interpretable machine learning model to predict and influence mortality of patients with heart failure warded in intensive care unit
- **K Nearest Neighbours using Shapley values from SHAP**



bicarbonate (in agreement to before)
- Numeric
- Higher value (red) will reduce odds of dying (negative SHAP values)

heart_rate (in agreement to before)
- Numeric
- Higher value (red) will increase odds of dying (positive SHAP values)

renal_failure (in agreement to before)
- Categorical
- Higher value (presence) will reduce odds of dying (negative SHAP values)

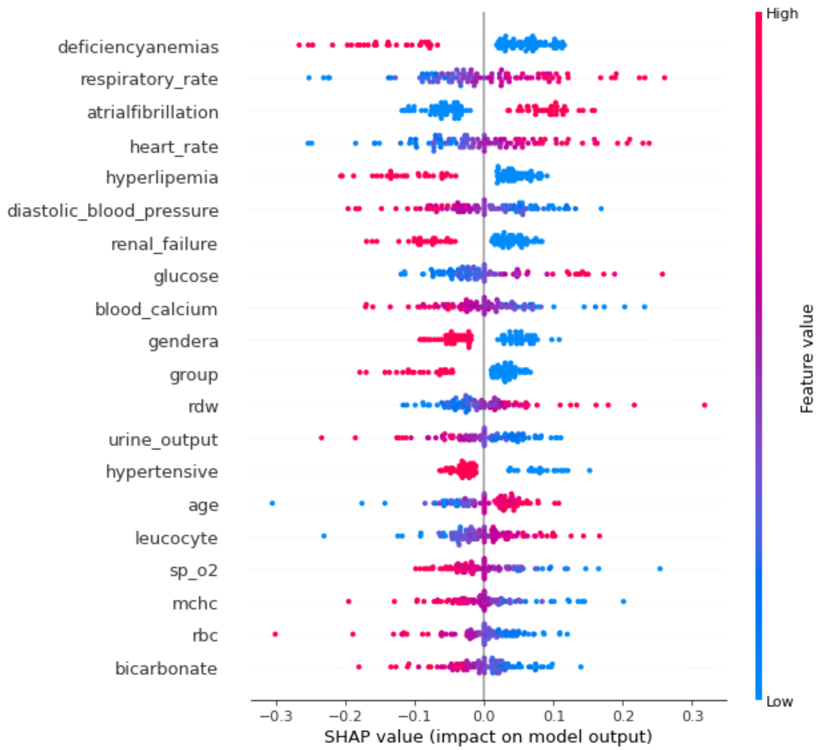
atrialfibrillation (in agreement to before):
- Categorical
- Higher value (presence) will increase odds of dying (positive SHAP values)

- Interpretable
  - Causes of death are understood

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- **Conclude on objective**
- Deployment workflow

- Develop interpretable machine learning model to predict and influence mortality of patients with heart failure warded in intensive care unit
- **Neural Network using Shapley values from SHAP**



bicarbonate (in agreement to before)
- Numeric
- Higher value (red) will reduce odds of dying (negative SHAP values)

heart_rate (in agreement to before)
- Numeric
- Higher value (red) will increase odds of dying (positive SHAP values)

renal_failure (in agreement to before)
- Categorical
- Higher value (presence) will reduce odds of dying (negative SHAP values)

atrialfibrillation (in agreement to before)
- Categorical
- Higher value (presence) will increase odds of dying (positive SHAP values)

- Interpretable
  - Causes of death are understood

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- **Conclude on objective**
- Deployment workflow

- Comparison of top important features in 3 machine learning models

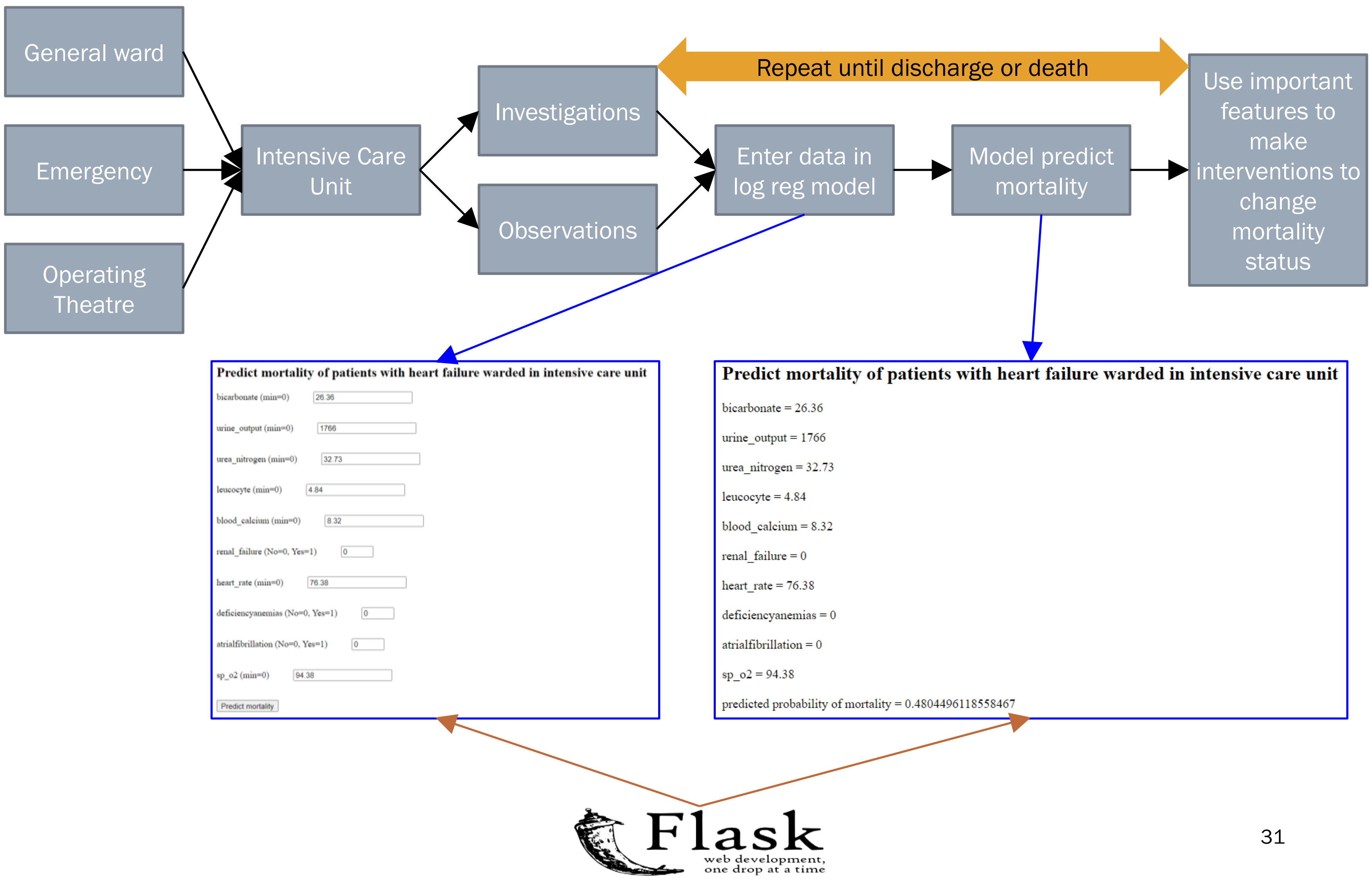
Logistic Regression	K Nearest Neighbours	Neural Network
Renal_failure	Bicarbonate	Deficiencyanemias
Bicarbonate	Systolic_blood_pressure	Respiratory_rate
Blood_calcium	Blood_calcium	Atrialfibrillation
Urine_output	Urine_output	Heart_rate
Deficiencyanemias	Heart_rate	Hyperlipemia
Sp_o2	Renal_failure	Diastolic_blood_pressure
Heart_rate	Diastolic_blood_pressure	Renal_failure
Leucocyte	Anion_gap	Glucose
Atrialfibrillation	Deficiencyanemias	Blood_calcium
Urea_nitrogen	Platelets	Gendera

- BLUE: Presence in 3 models

Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- **Deployment workflow**

- Workflow to execute data science solution:



Data Science Process

- Objective
- Collect
- Explore
- Engineer feature
- Build model
  - Logistic Regression
  - K Nearest Neighbours
  - Neural Network
- Conclude on objective
- Deployment workflow

THE END