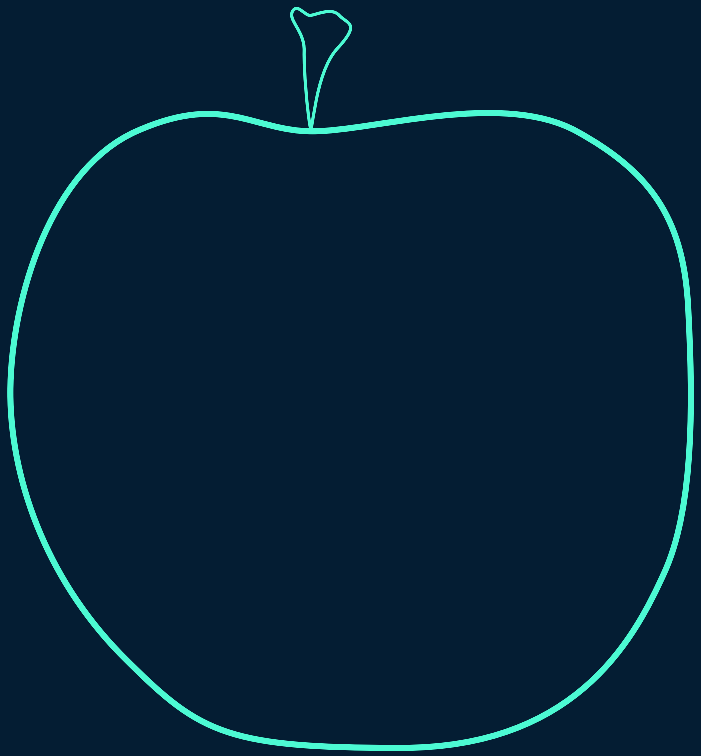



J. Gergely Hornyak

J. Gergely Hornyak
[Garry]

Binary Classification

J. Gergely Hornyak

Binary Classification

of apples and oranges

J. Gergely Hornyak

Binary Classification

of apples and oranges

J. Gergely Hornyak

30. 01. 2023.

Task:

Task:

- 10,000 pictures of oranges
- 1,000 pictures of apples

Figure 1: Apples and Oranges

Task:

- 10,000 pictures of oranges
- 1,000 pictures of apples

* premise: they are labelled

Task:

- 10,000 pictures of oranges
- 1,000 pictures of apples
- Classify images

A suitable machine learning model

A suitable machine learning model

- Decision Tree

A suitable machine learning model

- Decision Tree

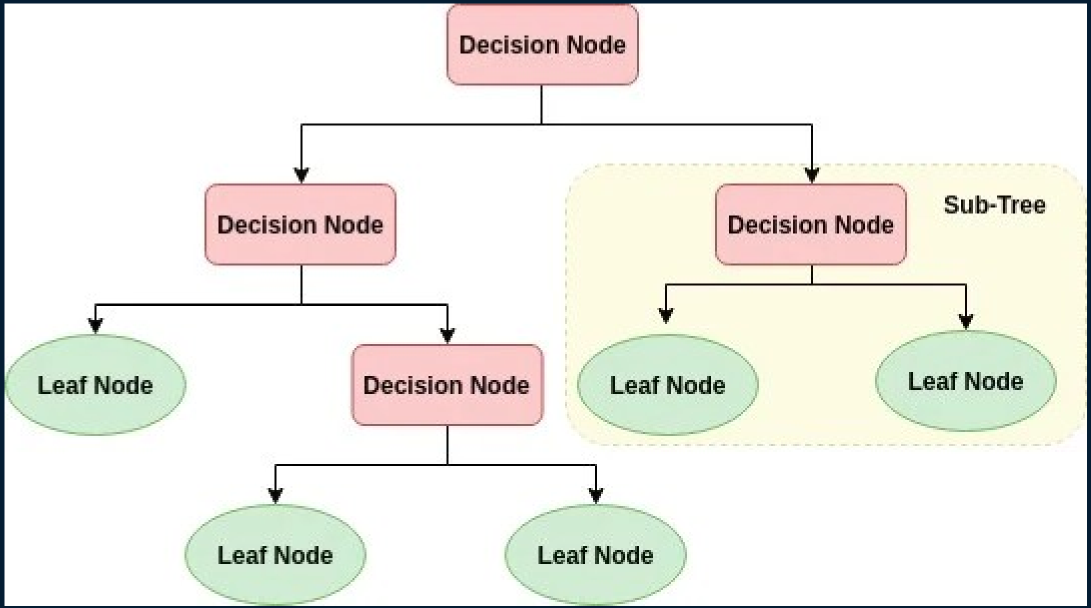


Figure 2: Decision Tree Nodes and Leaves

A suitable machine learning model

- Decision Tree

Figure 3: Plinko Game Board

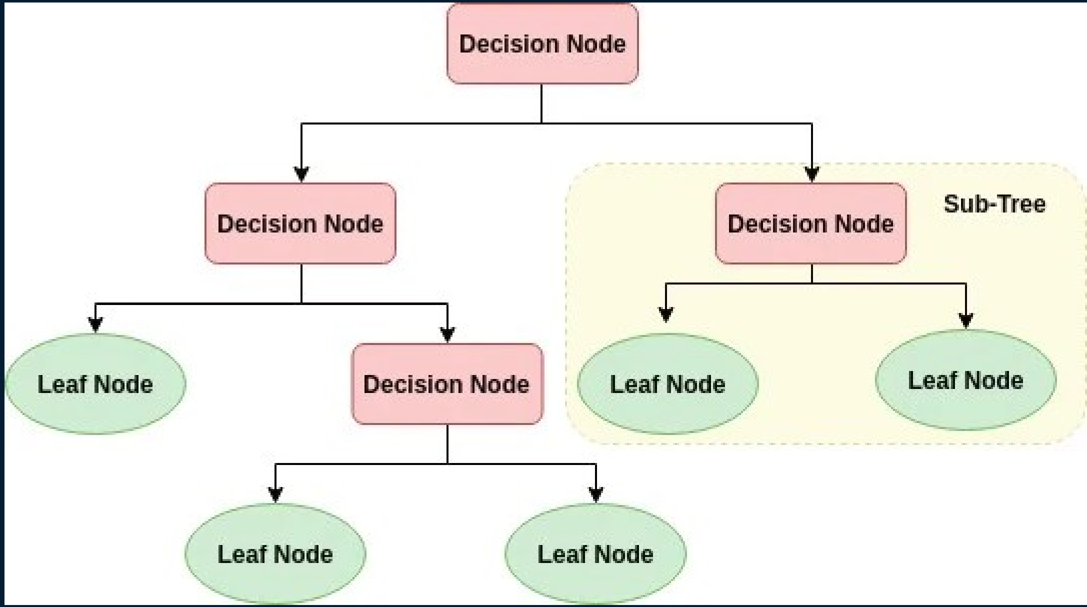


Figure 2: Decision Tree Nodes and Leaves

A suitable machine learning model

- Decision Tree
- Logistic Regression

A suitable machine learning model

- Logistic Regression

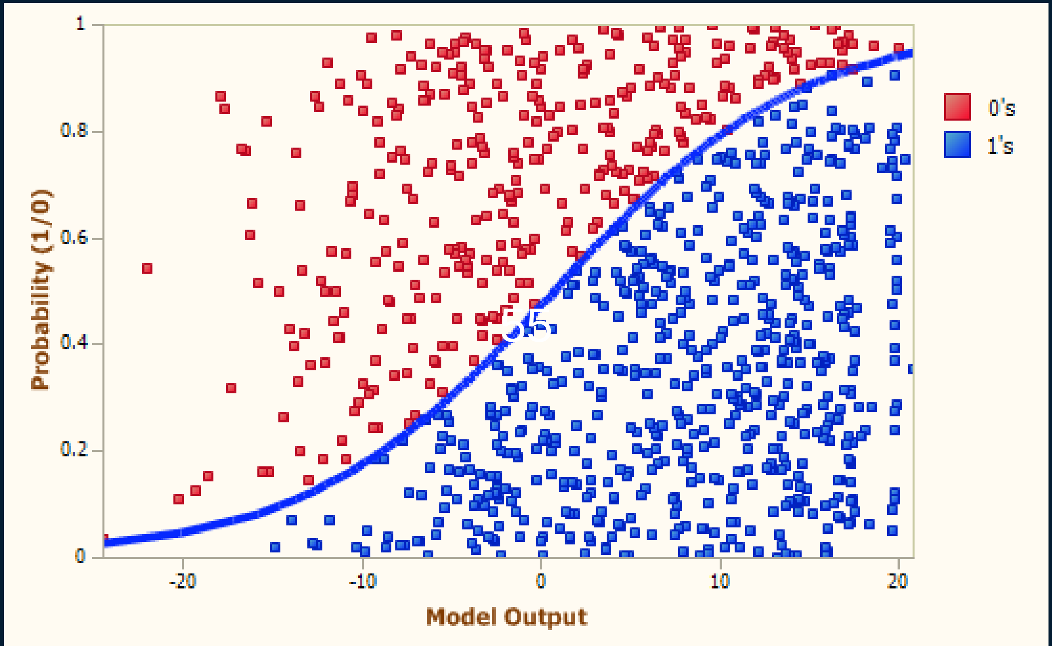


Figure 4: Logistic Regression Diagram

A suitable machine learning model

- Logistic Regression
- Decision Tree

A suitable machine learning model

- Logistic Regression
- Decision Tree

```
import sklearn library
function decision_tree(data, target_variable, feature_variables):
    if all observations have the same target variable value:
        return that target variable value as the prediction
    best_feature = find_best_feature(data, target_variable, feature_variables)
    tree = create_tree_node(best_feature)
    for value in unique values of best_feature:
        subset = data where best_feature is the value
        subtree = decision_tree(subset, target_variable, feature_variables)
        add subtree as a child of tree with value as the label
    return tree
```

Simple pipeline for classification

Data processing

1. Image processor
2. Feature extraction
3. Visualise data
4. Restructure data
5. Split data
6. Train model
7. Test model

Data processing

- OpenCV

Data processing

- OpenCV
- Tensorflow

Data processing

- OpenCV
- Tensorflow

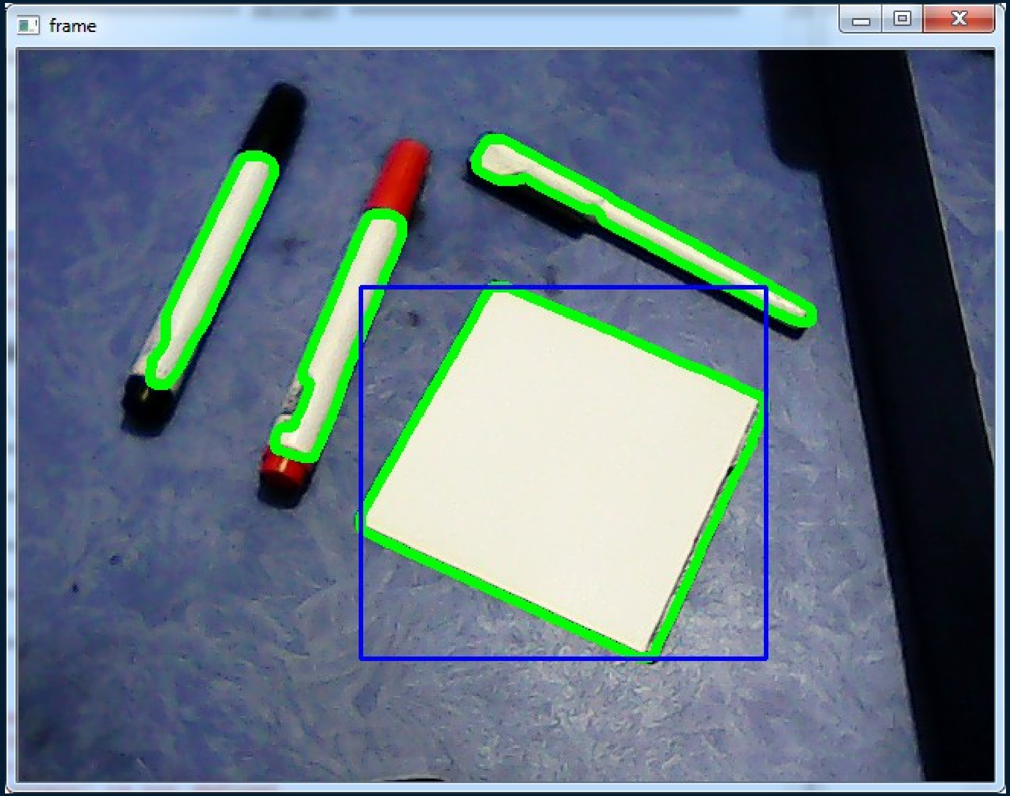


Figure 5: OpenCV Real Time Contours Detection

- Image processor
- Feature extraction
- Visualise data
- Restructure data
- Split data
- Train model
- Test model

Data processing

- OpenCV
- Tensorflow

Data processing

- OpenCV(2)

```
Import OpenCV library
function image_processing(image)
    image = Load image
    image = Convert image to grayscale
    image = Apply Gaussian blur to image
    image = Detect edges in image
    return image
```

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

[Data processing]

Feature extraction

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

[Data processing]

Feature extraction

- SIFT

Feature extraction

- SIFT

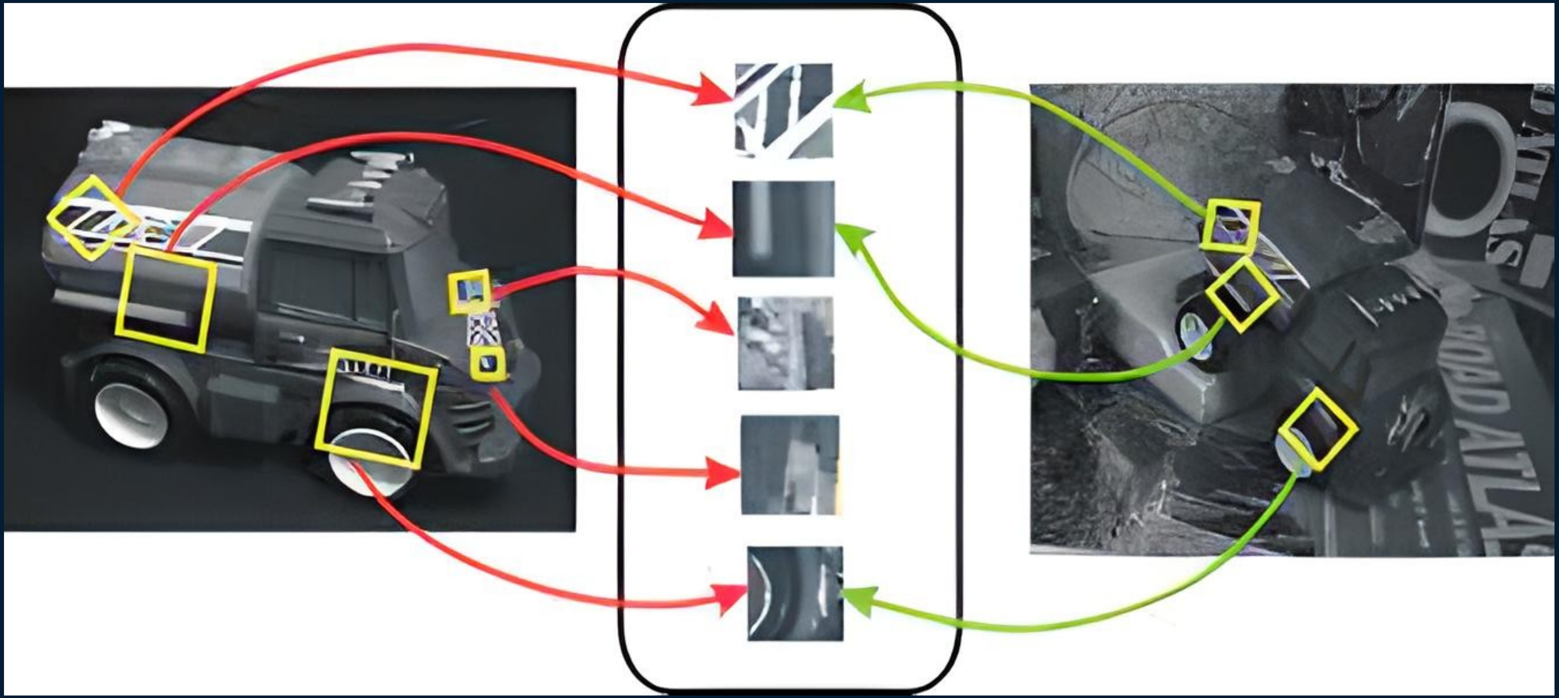


Figure 6: OpenCV’s Scale-Invariant Feature Transform

Image processor
Feature extraction
Visualise data
Restructure data
Split data
Train model
Test model

Feature extraction

- SIFT
- FAST

Feature extraction

- SIFT
- FAST

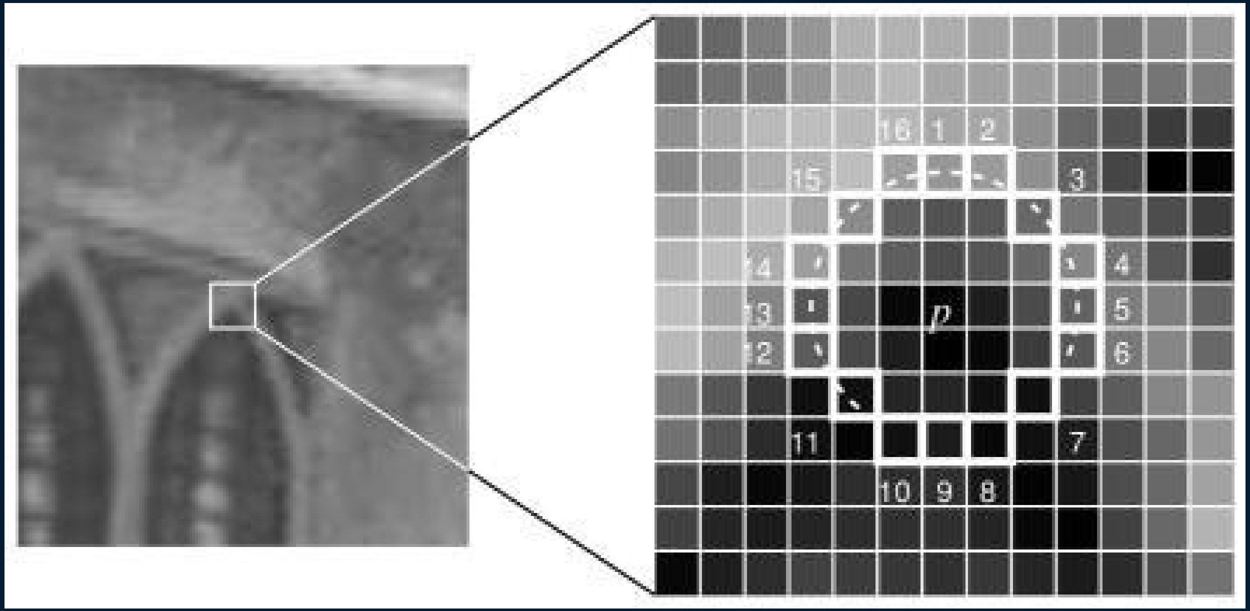


Figure 7: Features From Accelerated Segment Test Method

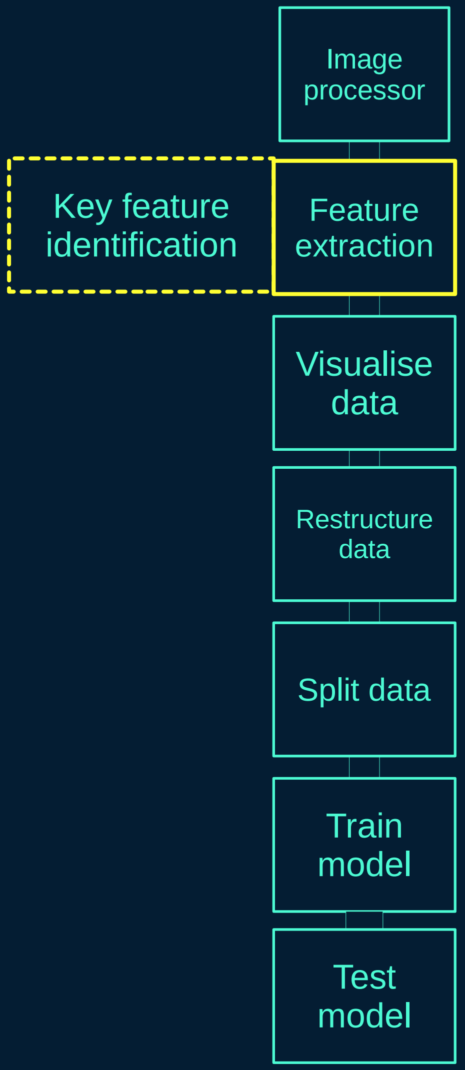
Feature extraction

- SIFT
- FAST

```
Import OpenCV library
function sift_feature_extraction(image)
    sift = Create SIFT object
    keypoints = Detect keypoints in image
    descriptors = Compute descriptors for keypoints
    return descriptors
```

[Data processing]

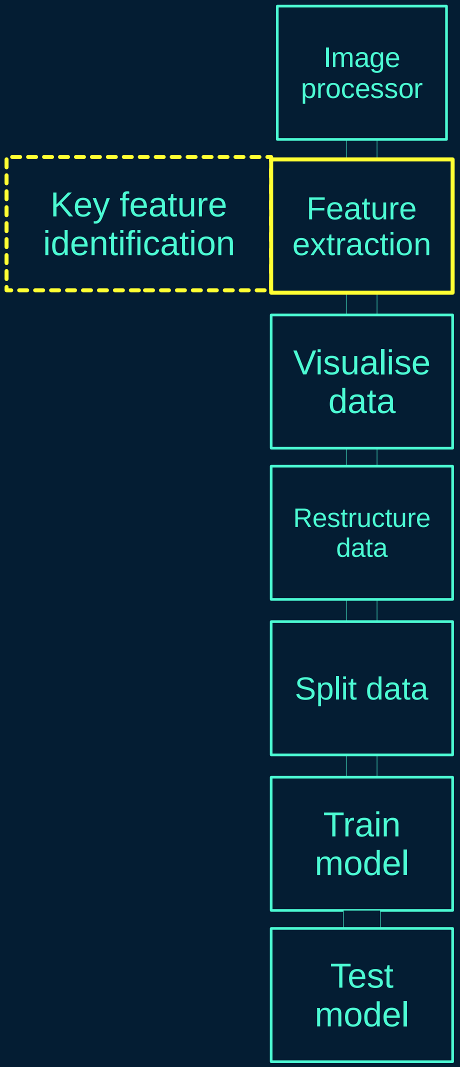
Key features



[Data processing]

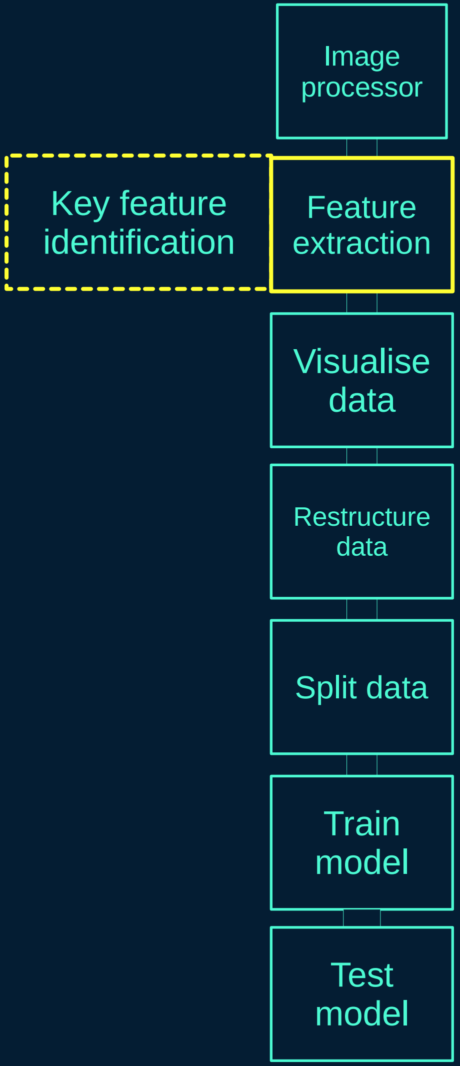
Key features

- Texture

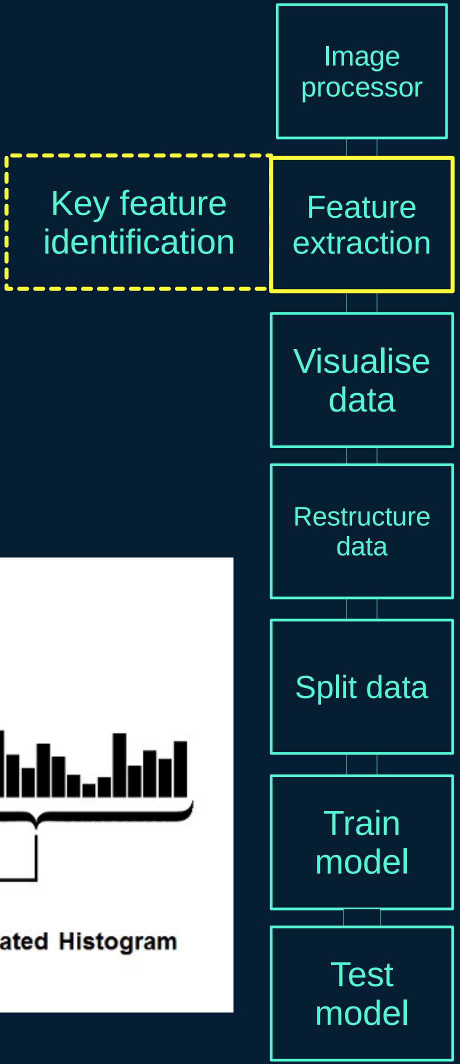


Key features

- Texture:
  - GLCM



Key features



- Texture:
  - GLCM

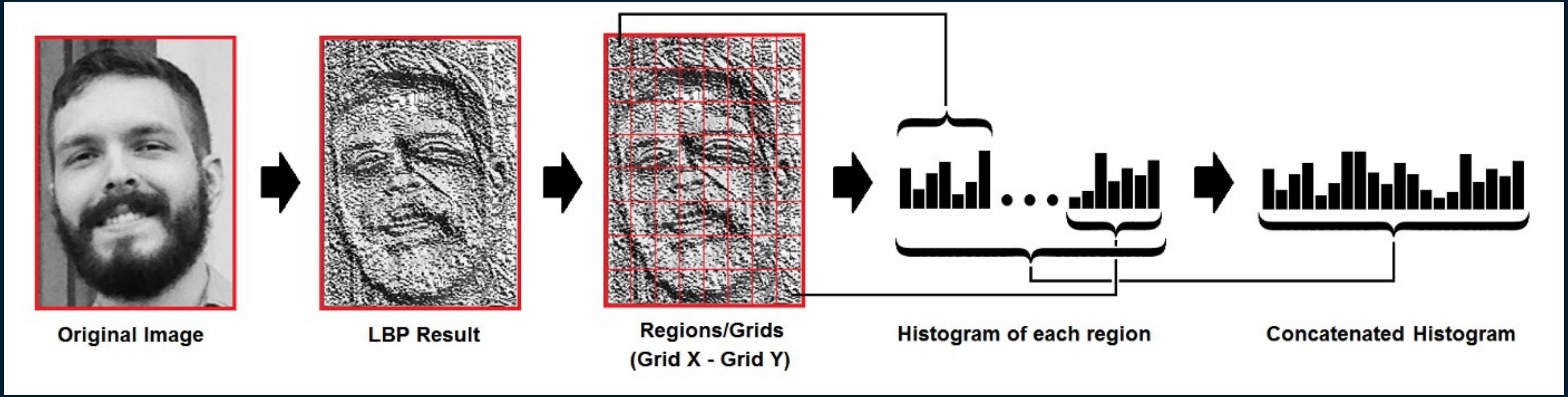
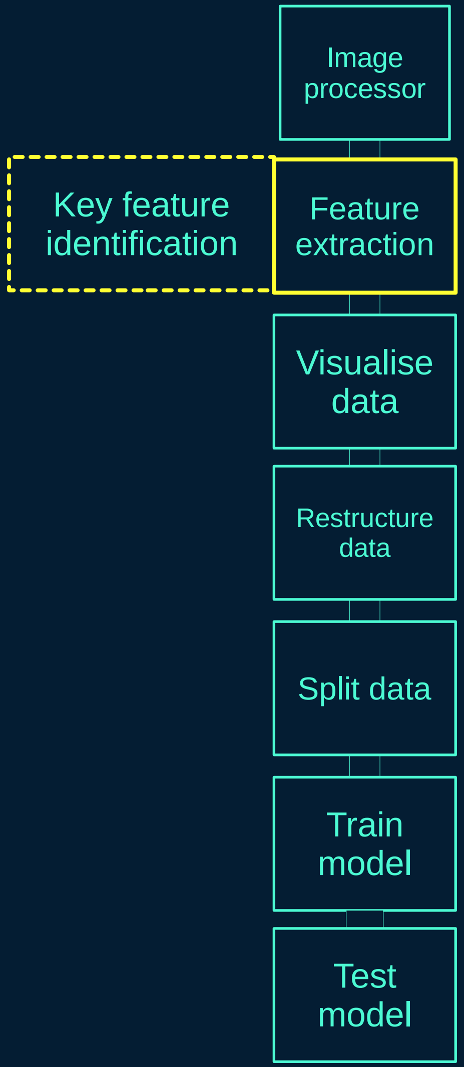


Figure 8: Gray-Level Co-Occurance Matrix Flowchart

Key features

- Texture:
  - GLCM
  - LBP



Key features

- Texture:
  - GLCM
  - LBP

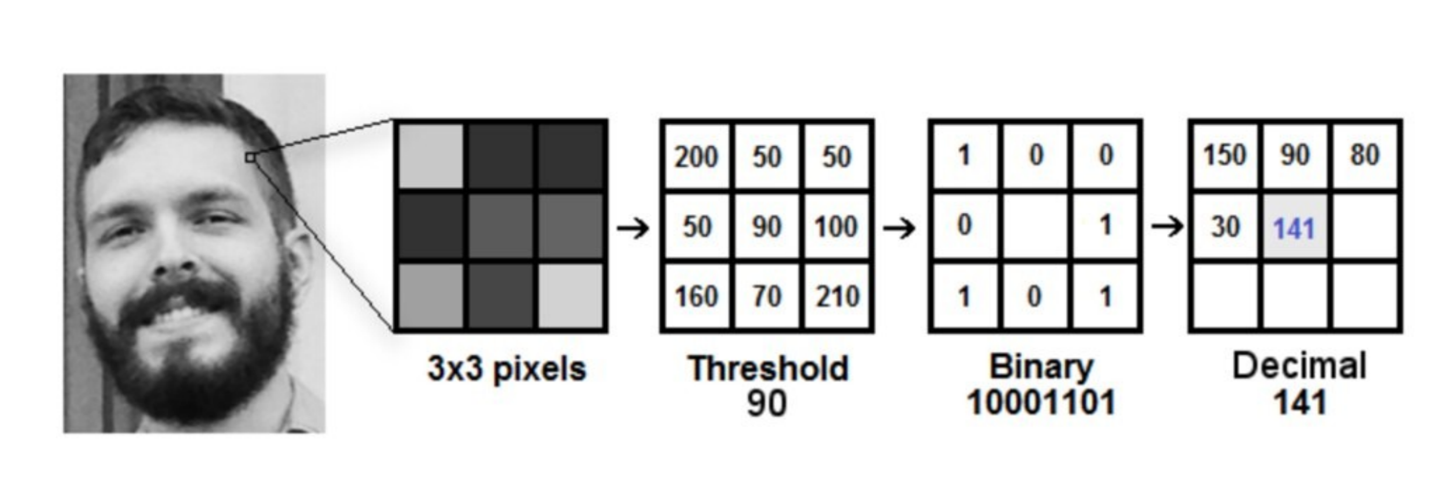
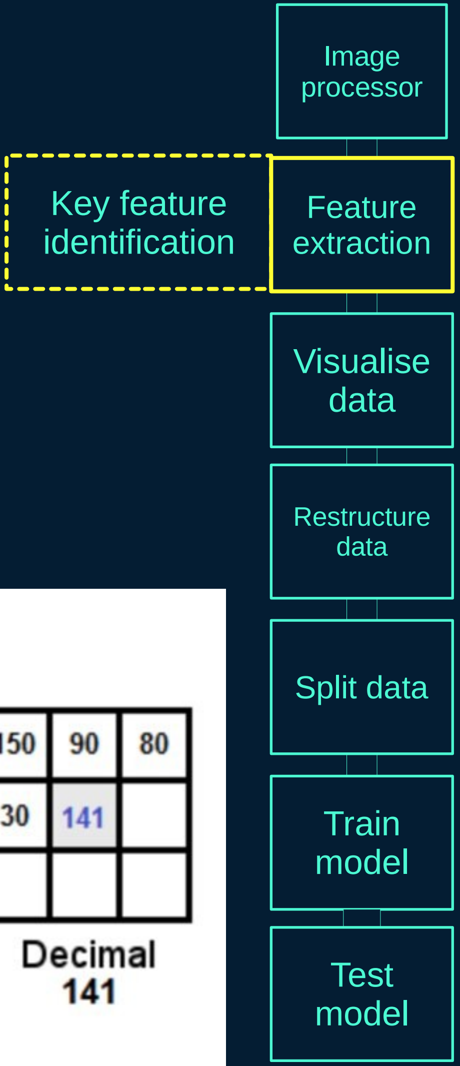


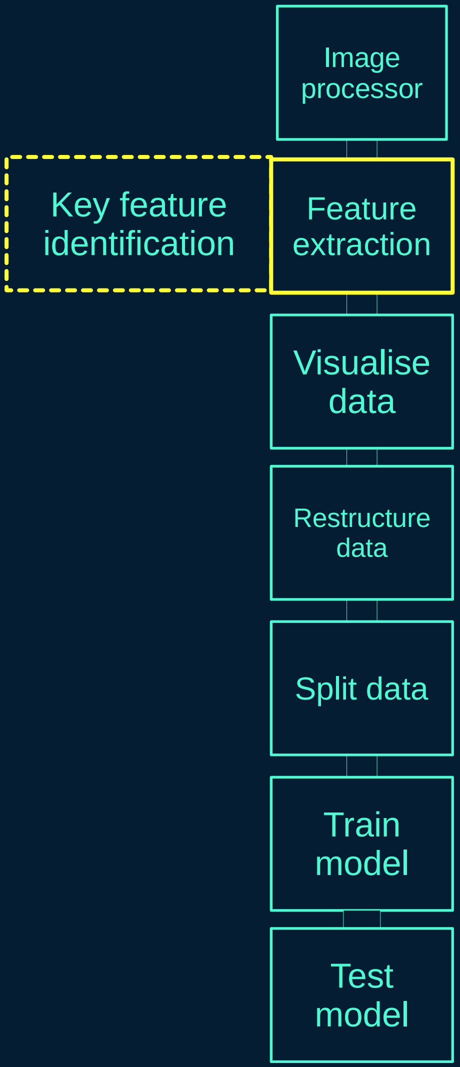
Figure 9: Local Binary Pattern Flowchart



[Data processing]

Key features

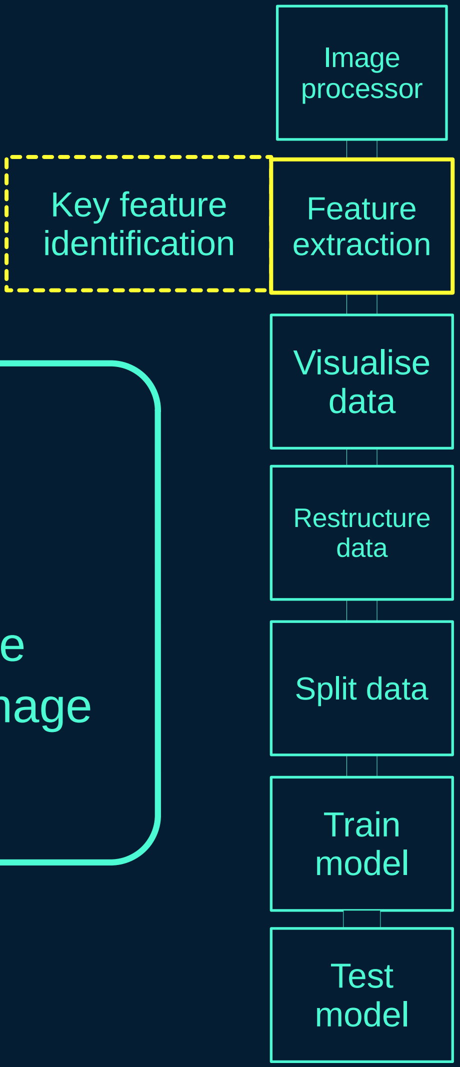
- Texture:
  - GLCM
  - LBP



Key features

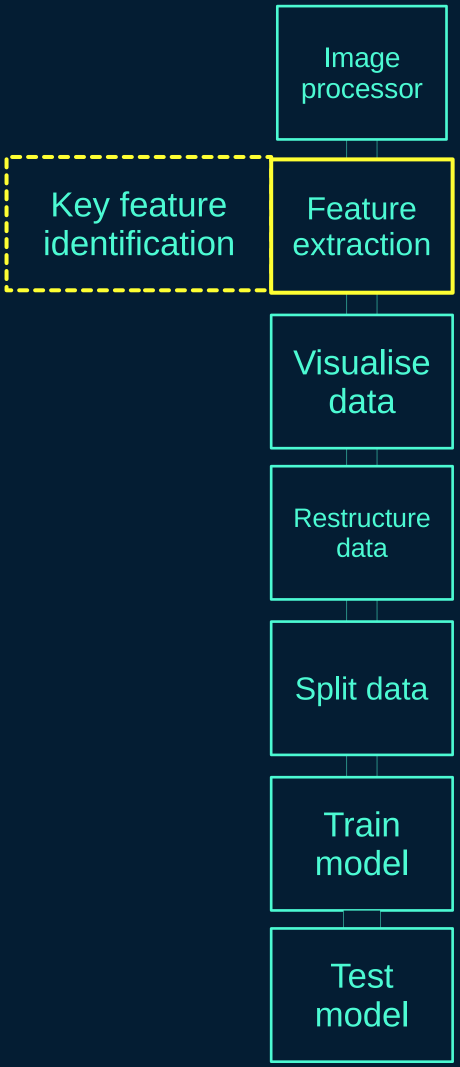
- Texture:
  - GLCM
  - LBP

```
Import OpenCV library
function LBP_feature_identification(image)
    image = Convert image to grayscale
    image = Apply LBP transformation to image
    histogram = Generate histogram of LBP image
    return histogram
```



Key features

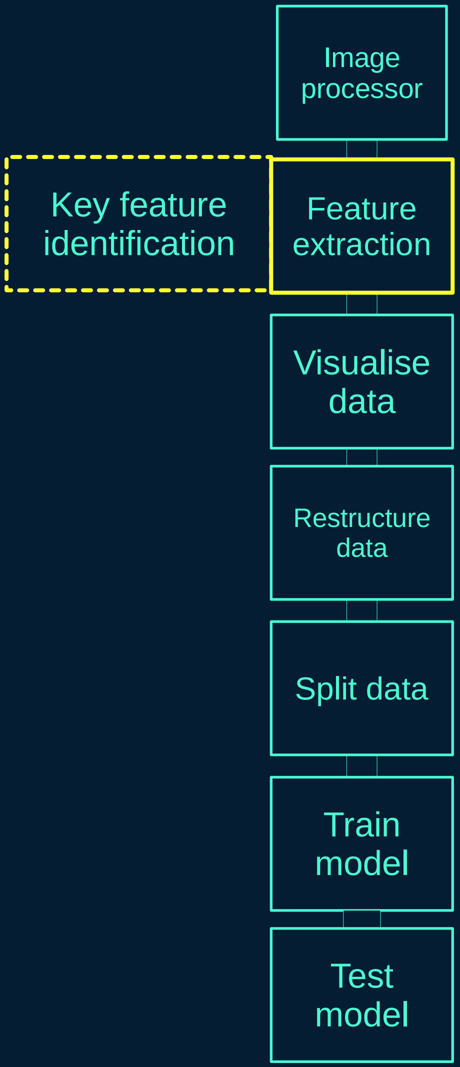
- Texture:
  - ~~GLCM~~
  - LBP



[Data processing]

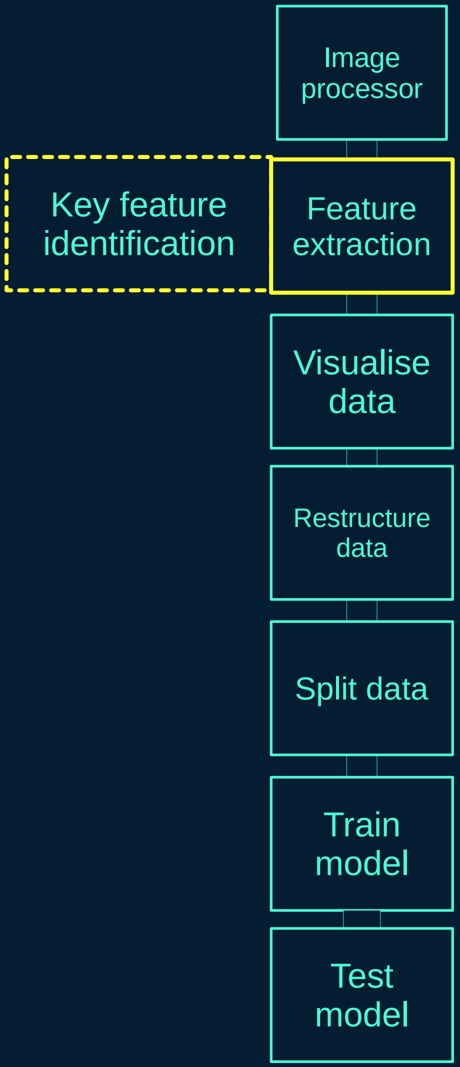
Key features

- Texture:
  - LBP
- Colour:



Key features

- Texture:
  - LBP
- Colour:
  - HSV



Key features

- Texture:
  - LBP
- Colour:
  - HSV

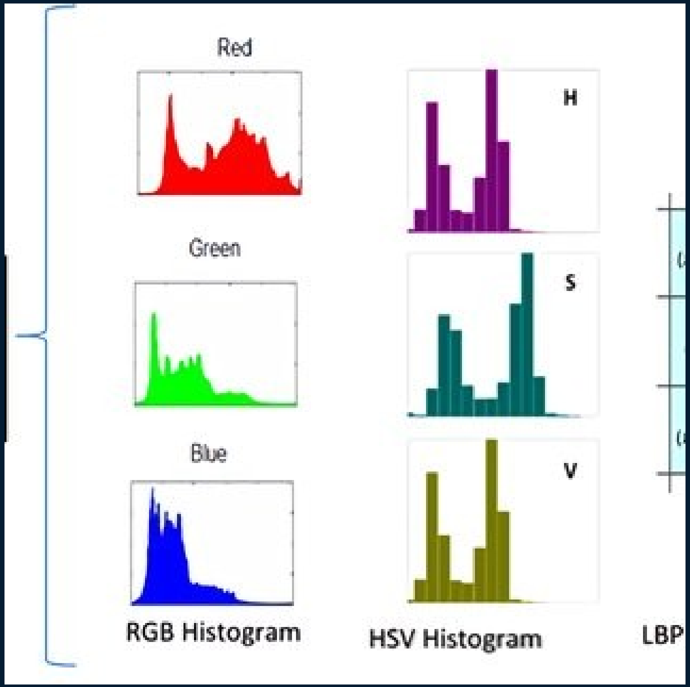


Figure 10: Hue – Saturation – Value Histograms

Image processor

Key feature identification

Feature extraction

Visualise data

Restructure data

Split data

Train model

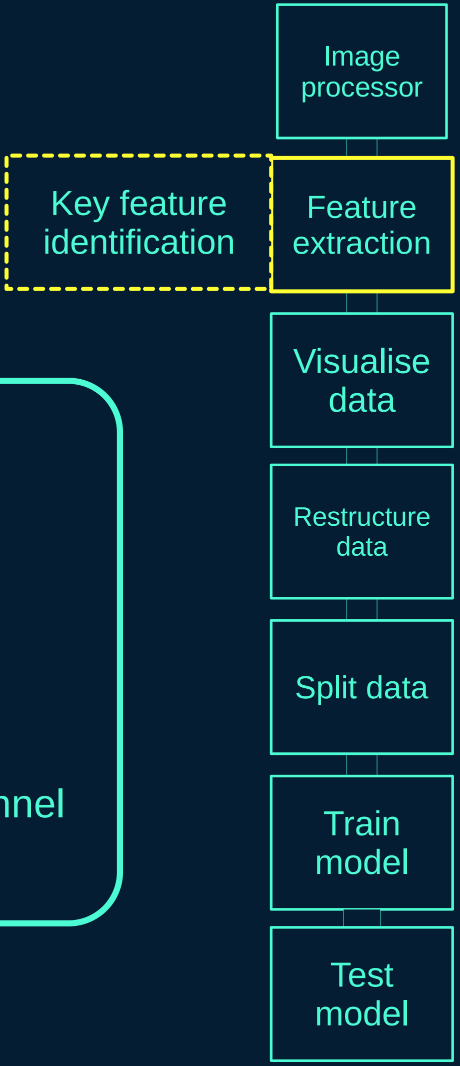
Test model

[Data processing]

Key features

- Texture:
  - LBP
- Colour:
  - HSV

```
Import OpenCV library
function HSV_feature_identification(image)
    image = Convert image to HSV color space
    hue, saturation, value = Split image into hue, /
                              saturation and value channels
    thresholded_hue = Threshold hue channel /
                              using a range of values
    features = Extract features from thresholded hue channel
    return features
```

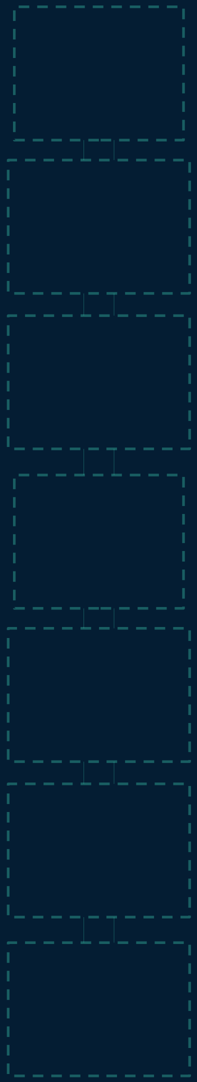


[Data processing]

Store features in database

Store features in database

- ID
- Colour:
- Texture:
- Result:



Store features in database

- ID
- Colour: RGB
- Texture: LBP -> local texture information
- Result: apple or orange

[Data processing]

Database analysis

Database analysis

- Imbalanced classification?

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

Database analysis

- Imbalanced classification?
- NaN values?

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix:

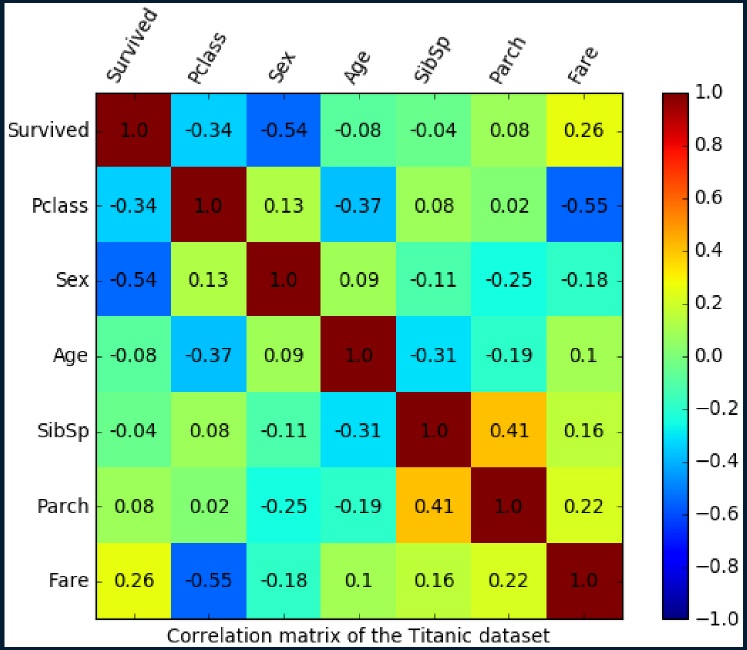


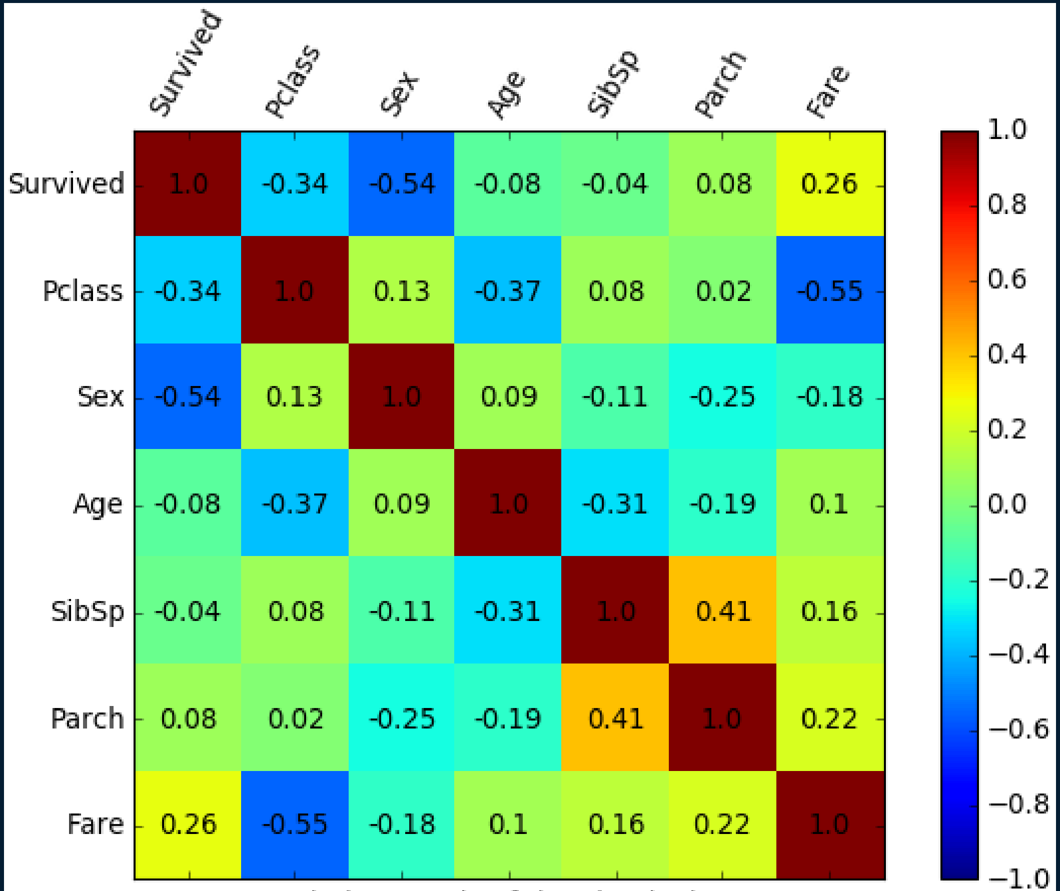
Figure 11: Titanic Dataset Correlation Matrix Heat map

- Image processor
- Feature extraction
- Visualise data
- Restructure data
- Split data
- Train model
- Test model

[Data processing]

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix:



Correlation matrix of the Titanic dataset

Image processor

Feature extraction

Visualise data

Restructure data

Split data

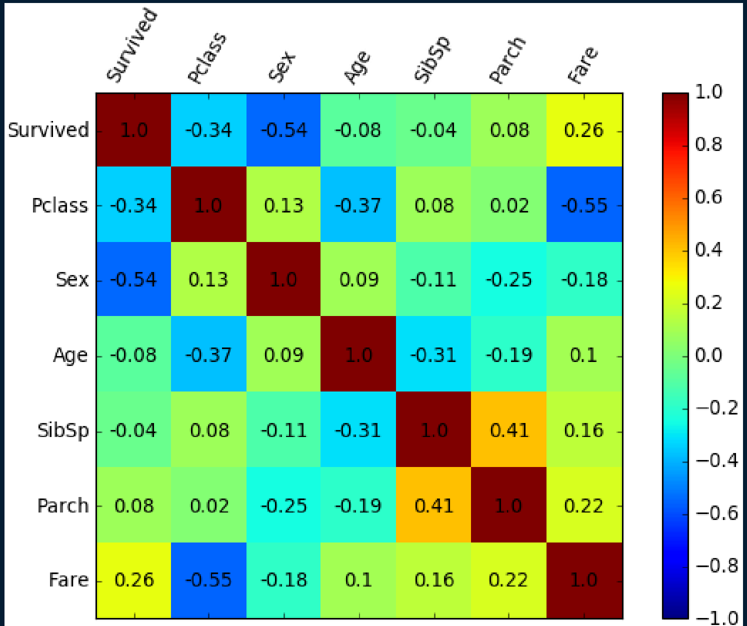
Train model

Test model

[Data processing]

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix:



Correlation matrix of the Titanic dataset

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:

Database analysis

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

[Data processing]

Database analysis + restructuring

- Imbalanced classification?
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Database analysis + restructuring

- Imbalanced classification!
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

Database analysis + restructuring

- Imbalanced classification!
  - Under-sampling
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Image processor
Feature extraction
Visualise data
Restructure data
Split data
Train model
Test model

Database analysis + restructuring

- Imbalanced classification:
  - Under-sampling

```
Import numpy and sklearn libraries
function under_sampling(data, target_class):
    majority_class, minority_class = Separate majority and minority class \
                                     from data based on target_class
    n_remove = Determine number of samples to remove from majority class \
               to balance with minority class
    majority_class_downsampled = Remove n_remove samples from majority class
    balanced_data = Concatenate minority_class and majority_class_downsampled
    return balanced_data
```

[Data processing]

Database analysis + restructuring

- Imbalanced classification!
  - Under-sampling
- NaN values?
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Database analysis + restructuring

- Imbalanced classification!
  - Under-sampling
- ~~NaN values~~
- Visualise with correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Image processor

Feature extraction

Visualise data

Restructure data

Split data

Train model

Test model

[Data processing]

Database analysis + restructuring

- Imbalanced classification:
  - Under-sampling
- Correlation matrix
- Feature importance determination:
  - Random Forest Regressor

Database analysis + restructuring

- Imbalanced classification:
  - Under-sampling
- Correlation matrix
- Feature importance determination:
  - Random Forest Regressor
- Categorise data / decompose

Database restructuring

- Normalisation

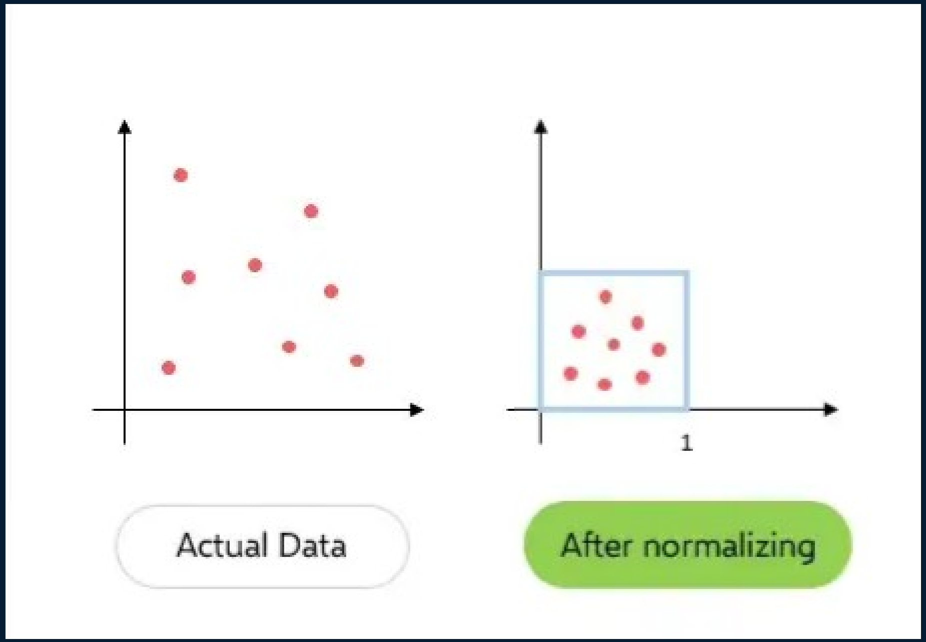


Figure 12: Normalised Data Before-After

Splitting data

[Data processing]

Splitting data

- train & test data

Splitting data

- train & test data
- + validation data

Splitting data

- training data: ~70%
- testing data: ~ 30%

Splitting data

- training data: 70%
- testing data: 30%

```
Import sklearn library
function split_data(data, target, test_size):
    data_train, data_test, target_train, target_test = Split data and target /
                                into training and testing sets using test_size
    return data_train, data_test, target_train, target_test
```

Assessment of performance

Assessment of performance

- Train model

Assessment of performance

- Train model

```
import sklearn library
function train_decision_tree_model(data_train, target_train):
    dt_model = Create decision tree model
    Fit the model using data_train and target_train
    return dt_model
```

Assessment of performance

- Train model
- R2 score

Image processor
Feature extraction
Visualise data
Restructure data
Split data
Train model
Test model

Assessment of performance

- Train model
- R2 score
- Confusion matrix

Assessment of performance

- Train model
- R2 score
- Confusion matrix

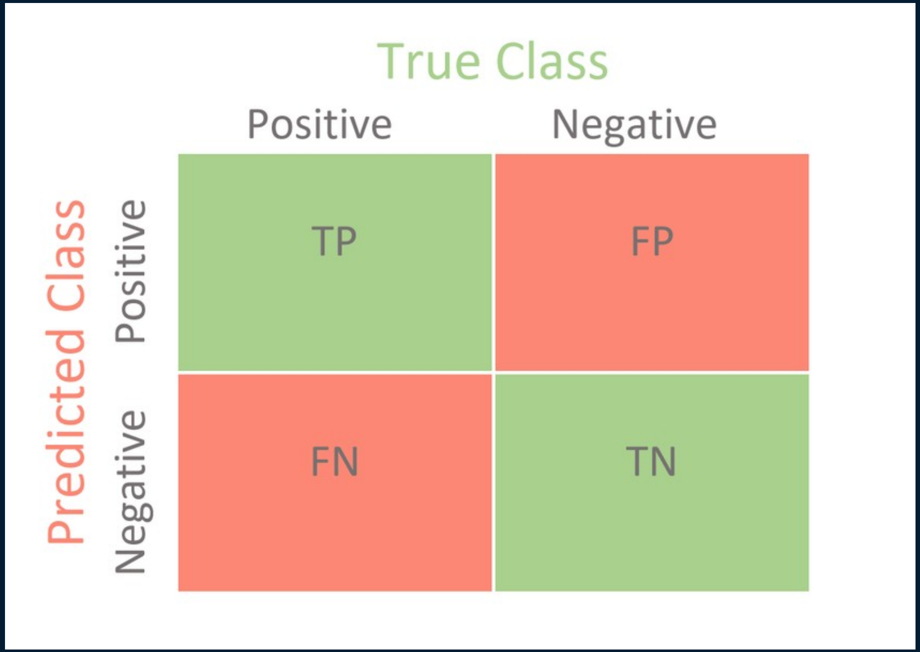
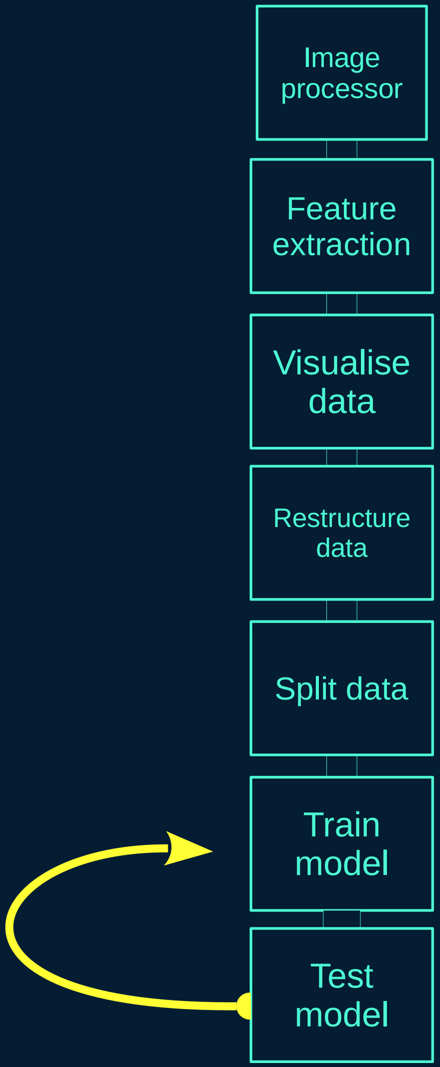


Figure 13: Confusion Matrix Example

[sidenote]

Fine-tuning

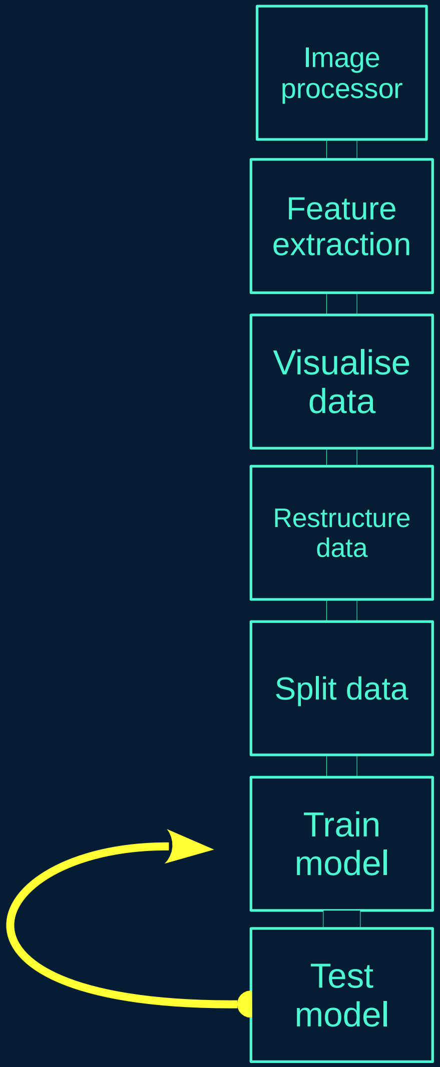
- Improve pre-trained model



[sidenote]

Fine-tuning

- Improve pre-trained model
- Reuse

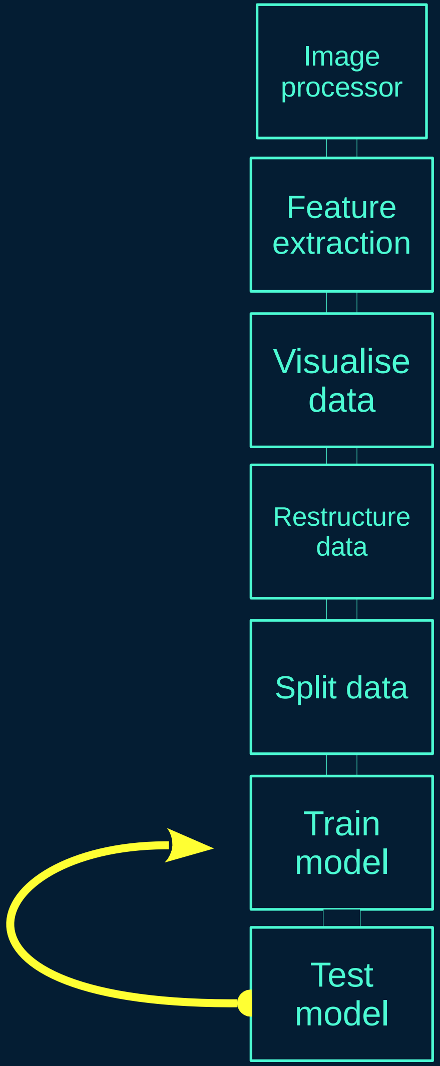


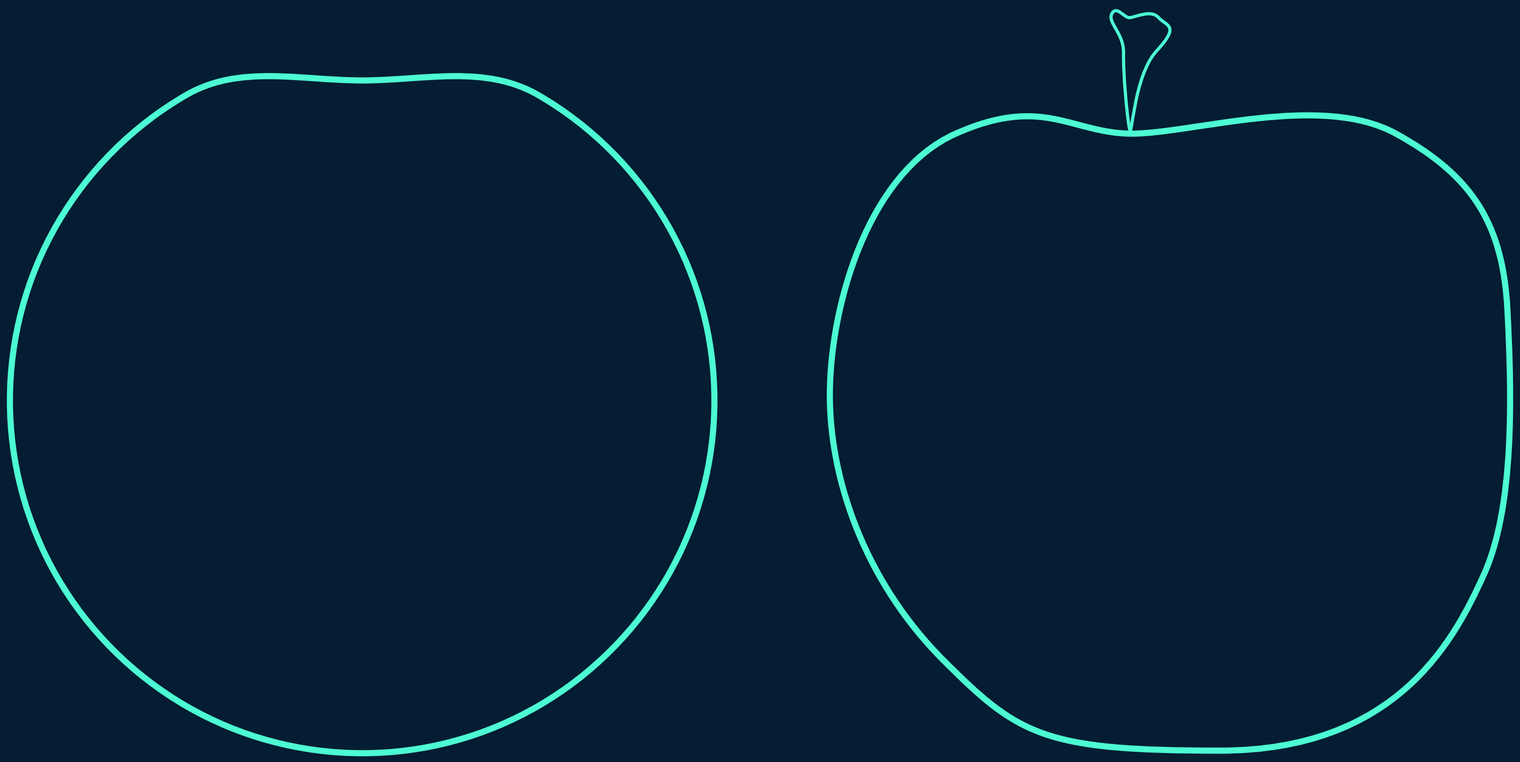
[sidenote]

Fine-tuning

- Improve pre-trained model
- Reuse

```
function fine_tune_model(model, X_train, y_train, X_val, y_val, /
        max_depth=None, min_samples_leaf=1, min_samples_split=2):
    model.set_params(max_depth=max_depth, /
            min_samples_leaf=min_samples_leaf, /
            min_samples_split=min_samples_split)
    model.fit(X_train, y_train)
    val_acc = model.score(X_val, y_val)
    return model, val_acc
```





END OF PRESENTATION PART

References

Kargas, N., & Sidiropoulos, N. D. (2020, November). Supervised Learning via Ensemble Tensor Completion. In 2020 54th Asilomar Conference on Signals, Systems,and Computers (pp. 196-199). IEEE.

Nasteski, V. (2017). An overview of the supervised machine learning methods. Horizons.b, 4, 51-62.

Niculescu-Mizil, A., & Caruana, R. (2005, August). Predicting good probabilities with supervised learning. In Proceedings of the 22nd international conference on Machine learning (pp. 625-632).

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. arXiv preprint arXiv:1811.12808.n

Thanks for my supervisor – Mark Elshaw - for perusing this presentation.

References

Fig. 1: https://media.npr.org/assets/img/2014/03/24/oranges-vs-apples-78eb1373a9239991a842766c3d5f203cf0d50cb1.jpg

Fig. 2: https://res.cloudinary.com/dyd911kmh/image/upload/f_auto,q_auto:best/v1545934190/1_r5ikdb.png

Fig. 3: https://i.etsystatic.com/7103629/r/il/e9acff/2049460861/il_570xN.2049460861_fehc.jpg

Fig. 4: https://1394217531-files.gitbook.io/~/files/v0/b/gitbook-legacy-files/o/assets%2F-LvBP1svpACTB1R1x_U4%2F-Lw70vAIGPfRR1AjprLi%2F-LwAVc1EdfmPMge5dlYC%2Fimage.png?alt=media&token=d72e3231-0d64-4bb7-9e4c-20577940763d

Fig. 5: https://i1.wp.com/www.fypsolutions.com/wp-content/uploads/2020/04/detectcontours-opencv-python.png

Fig. 6: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSgphOh2kydA_hYasIsM0uUxOAc6KW74Y4np7Ii3a3hG80ou5lvvvpsKTlkSao5r3ZDPsg&usqp=CAU

Fig. 7: https://miro.medium.com/max/443/0*iB26EPO33F4LSig3.jpg

Fig. 8: https://miro.medium.com/max/1400/1*-cyqWPcas3CXp4O2O7xPpg.png

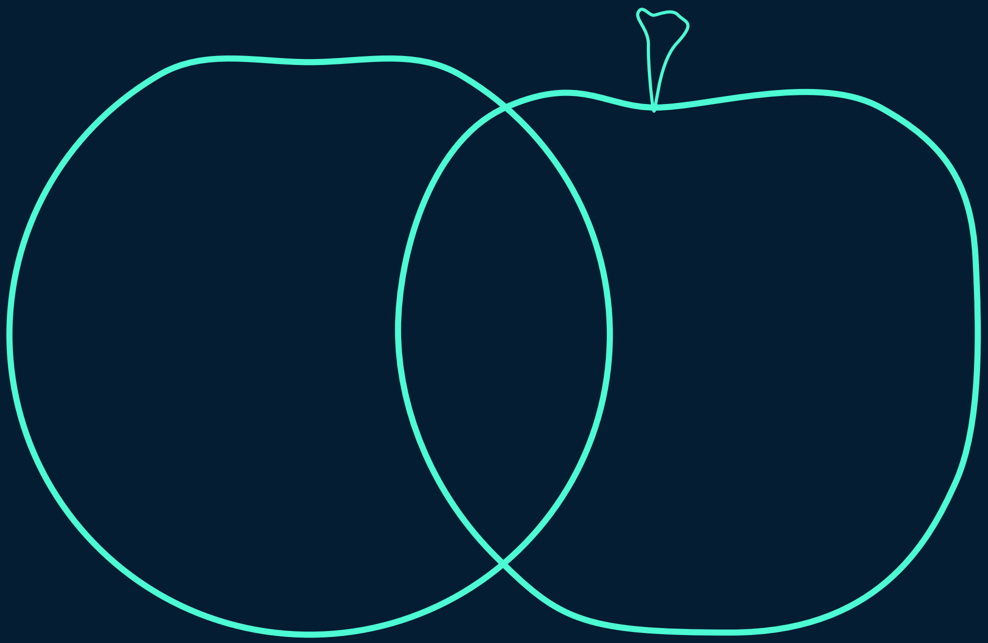
Fig. 9: https://farm5.staticflickr.com/4904/45927798041_307bae04be_b.jpg

Fig. 10: https://www.researchgate.net/publication/352060130/figure/fig1/AS:1030241258577927@1622640065528/Illustration-of-HSV-histograms-RGB-histograms-and-LBP-feature-extraction_Q320.jpg

Fig. 11: https://www.vertica.com/wp-content/uploads/2019/09/corr_matrix_Titanic.png

Fig. 12: https://miro.medium.com/max/744/1*HW7-kYjj6RKwrO-5WTLkDA.png

Fig. 13: https://www.researchgate.net/publication/350487701/figure/fig1/AS:1007018756280322@1617103389957/Confusion-Matrix-for-Binary-Classification-7.ppm



Questions & Answers

Thank you for your attention!