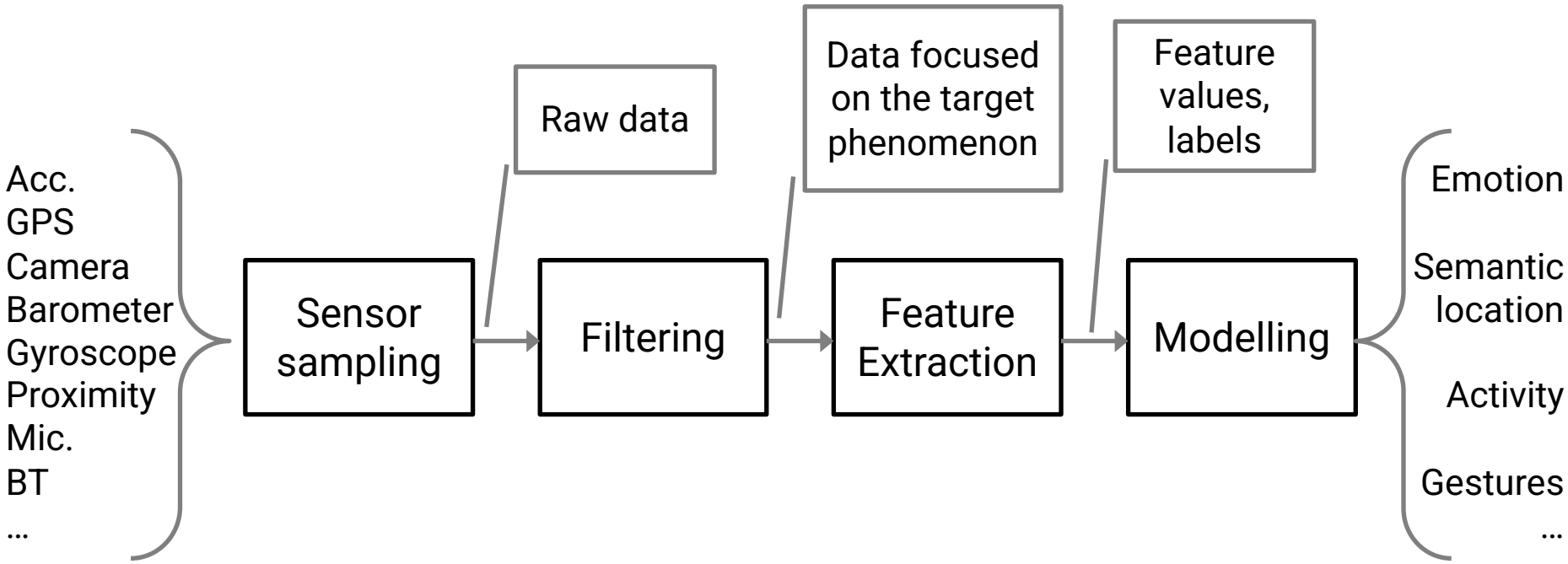
Mobile Sensing: Learning from Sensor Data

Master studies, Winter 2021/2022

Dr Veljko Pejović
Veljko.Pejovic@fri.uni-lj.si

University *of Ljubljana*
Faculty *of Computer and Information Science*

Sensing and Learning Pipeline



Machine Learning Basics

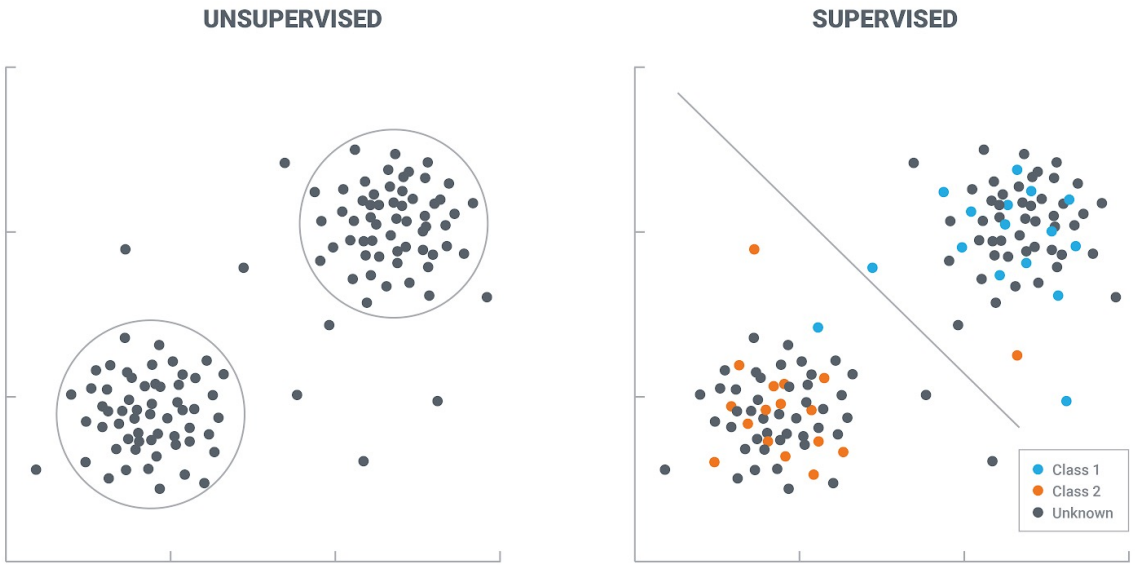
- Machine learning (ML)
  - Build mathematical models explaining a higher-level concept based on the previously collected training data
  - Predict/infer the higher-level concepts from the newly-collected data using the above models

Machine Learning Basics

- (Some) types of ML
  - Supervised learning: training data contains both the input values as well as the output (labels)
    - The model is built to predict the labels from the future non-labelled data that contains only inputs
    - E.g. a model that predicts physical activity (labels: walk, run, sit) from accelerometer data
  - Unsupervised learning: training data contains unlabelled data
    - The model finds structure in data
    - E.g. a model that finds top five commonly visited locations in GPS data

Machine Learning Basics

- Supervised vs Unsupervised learning



Source: https://lawtomated.com/supervised-vs-unsupervised-learning-which-is-better/

Machine Learning in Mobile Sensing

- The goal is to obtain a mathematical description connecting sensed signals and a high-level concept
  - e.g. accelerometer data and a user's physical activity
- If the high-level concept has discrete values, we use classification
  - e.g. activity recognition ("running", "sitting", "walking")
- If the high-level concept is described by continuous values, we use regression
  - e.g. depression level (0-24)

Running example – Activity Recognition

- Goal: activities from accelerometer data
- Approach: classifier for a well-defined set of activity types
- Steps:
  - Data collection
  - Data inspection
  - Feature engineering
  - Classifier construction
  - Model evaluation

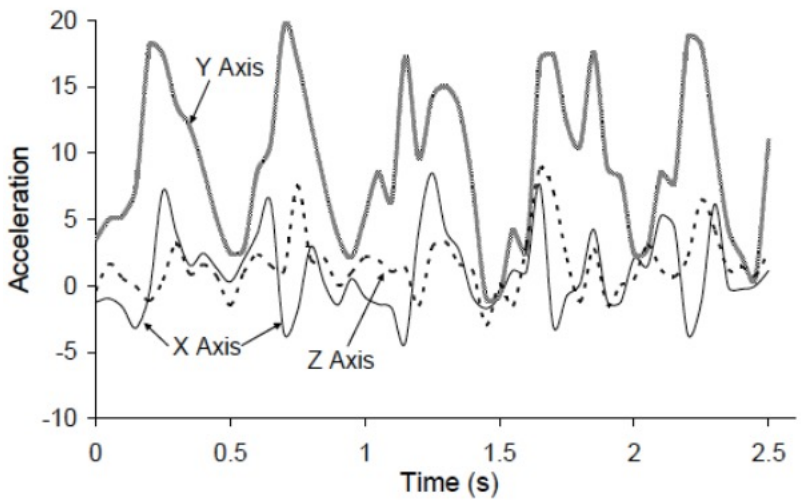
Data Collection

- Prerequisite: obtain Institutional Review Board clearance and recruit volunteers
- Labelling: sample sensors while the participants are performing each of the activities and assign the correct label (ground truth) to each sample
  - Equal amounts of data for each activity
  - Equal amounts of data from each user
  - Varying device placement (in a pocket, bag, etc.)
- The resulting dataset should be split into training, validation, and test dataset

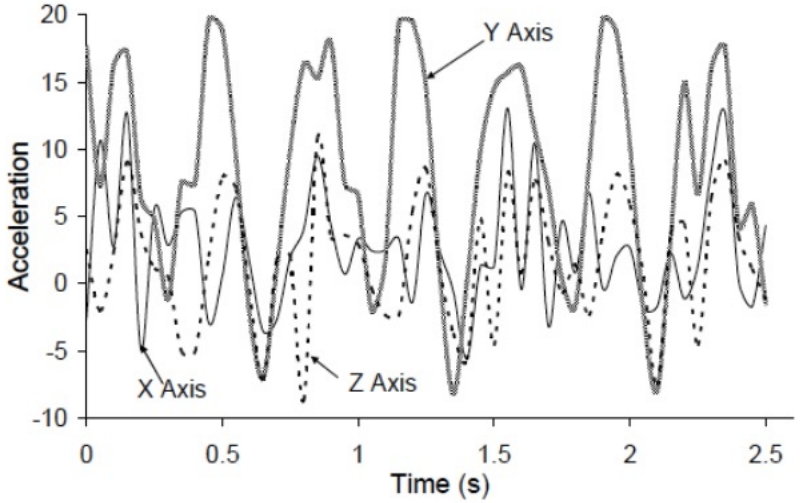
TIP: sometimes it is difficult to get a balanced dataset (e.g. walking vs falls). Under- and over-sampling might help

Data Inspection

- Visualisation
  - Iteratively (before and after filtering, outlier removal)
  - Different classes
  - Different scales (e.g. duration intervals)



(a) Walking

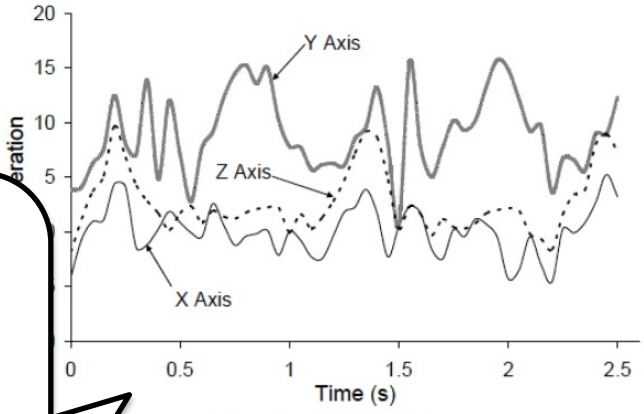


(b) Jogging

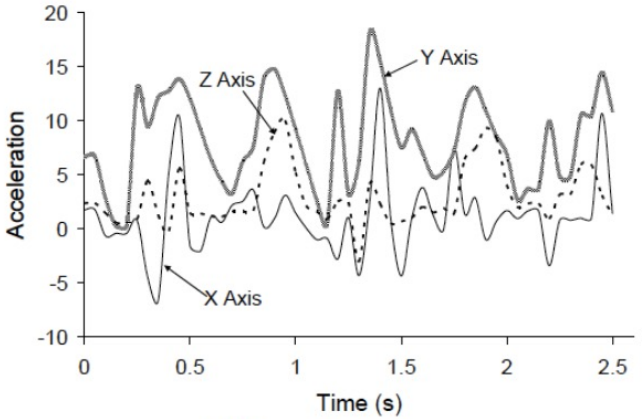
Data Inspection

- Visualisation

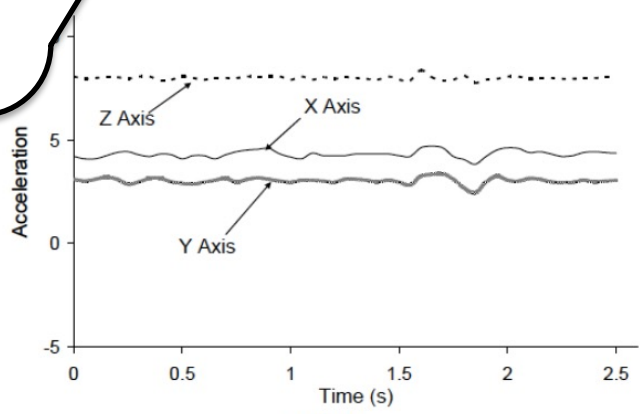
TIP: Visualisation helps you pinpoint discriminating characteristics of different classes.



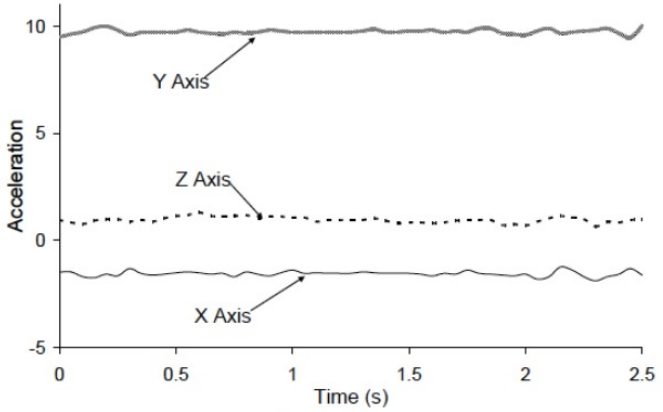
(c) Ascending Stairs



(d) Descending Stairs



(e) Sitting



(f) Standing

Feature Engineering

- Summary representations of the data
  - Time domain features: mean, variance, mean crossing rate (MCR), minimum, maximum, etc.
  - Frequency domain features: dominant frequency, power in different frequency bands, entropy, etc.
  - Domain specific
    - Search the related work to get ideas for new ones or to identify proven features in the domain you are tackling
  - Ensure consistent treatment of the data:
    - Equal time-window for each sampling segment, etc.

TIP: “Anticipatory Mobile Computing: A Survey of the State of the Art and Research Challenges” contains an overview of features used in different domains. Find it on Ucilnica!

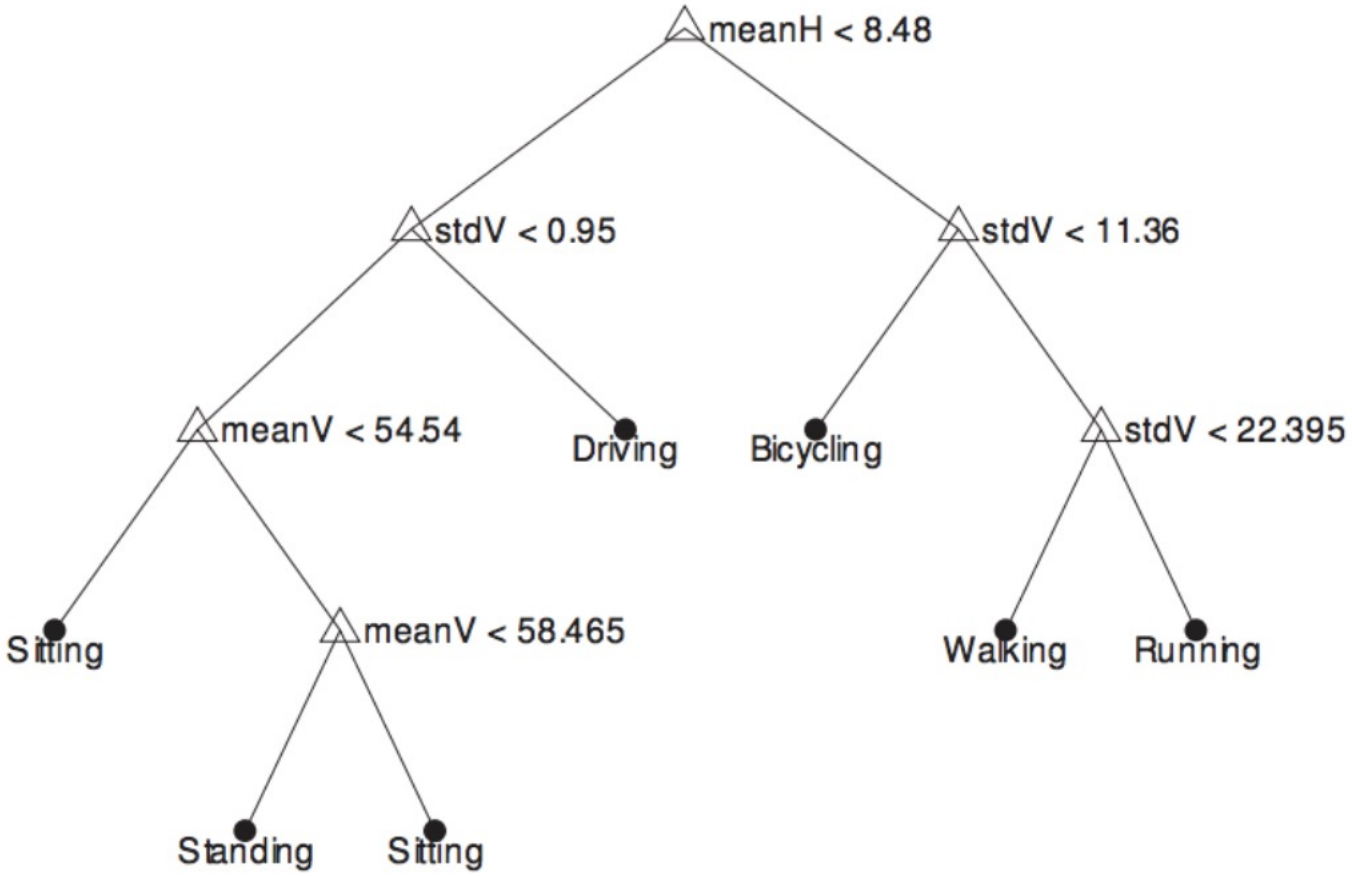
Classification/Regression

- Classification algorithms:
  - Tree-based
  - Bayesian
  - Ensemble methods
  - etc.
- Regression algorithms:
  - Linear and logistic regression
  - Multilevel models
  - etc.

For this class, you must know how a simple tree-based, Naïve Bayesian, and linear regression algorithms work!

Tree-Based Classification

- Example: distinguish between driving, sitting, standing, walking, jogging, cycling



Tree-Based Classification

- To construct a tree:
  - Find the feature that splits the data in the best possible manner – what does "best" mean?
  - Entropy (at a node *t*):

$$H(t) = -\sum_{i=0}^{c-1} p(i \mid t) \log_2 p(i \mid t)$$

More "pure" the nodes are after the split, the lower the entropy

  - Information gain (when split on a certain feature):

$$\Delta = H(S) - \sum_{t=0}^{T} \frac{N(t)}{N(S)} H(t)$$

  - Split on a feature that yields the highest Δ

Classifier Evaluation – Data Splitting

- Classifier must be built on one (training set) and evaluated on a separate dataset (test set)
  - You know the labels for the test set, but the classifier "sees" only the feature values and infers the labels
- Splitting datasets:
  - Holdout: x% of the data is held out for testing, (100-x)% of the data is used for training
  - N-fold cross validation: data divided into N equal subsets, one fold held out for testing, the rest used for training; calculate performance metrics and repeat N times, each time with a different subset held out

Classifier Evaluation – Data Splitting

- Validation set
  - If your classifier has a number of tunable parameters, further split the training set and use one part as a validation set
  - Tune parameters, test performance on the validation set; repeat until the best tuning is found
- Think before splitting the dataset!
  - Is your model built on data randomly sampled from a wider time period?
    - If the target property changes over time, how will your model work once you release it?
  - Does your training set include data from all users?
    - How will your model behave when a new user joins in?

Classifier Evaluation – Performance Metrics

- Accuracy:
  - Num of instances with correctly predicted labels/ Total number of instances
- Precision:
  - Accuracy of predicting a certain class
  - TP/(TP+FP)
- Recall
  - Ability to select instances of a certain class
  - TP/(TP+FN)
- F-measure
  - 2/(1/precision + 1/recall)

Classifier Evaluation – Performance Metrics

- Confusion matrix:
  - Rows: actual
  - Columns: predictions

Correct predictions are on the diagonal

	Easy	Medium	Difficult
Easy	158	101	65
Medium	98	163	63
Difficult	69	91	164
Precision	**49%**	**46%**	**56%**
Recall	**49%**	**50%**	**51%**
F1	**49%**	**48%**	**53%**
Accuracy		**51%**	

Classifier Evaluation – Performance Metrics

- Are the results good or not?
  - 90% accuracy means nothing
  - 90% accuracy is good if the user is equally likely to run, walk, sit, stand, go up/down the stairs
  - 90% is bad if the user is anyways sitting 90% of the time
- Always compare to the baseline!
- Baseline:
  - Mean value for regression
  - Majority class for classification

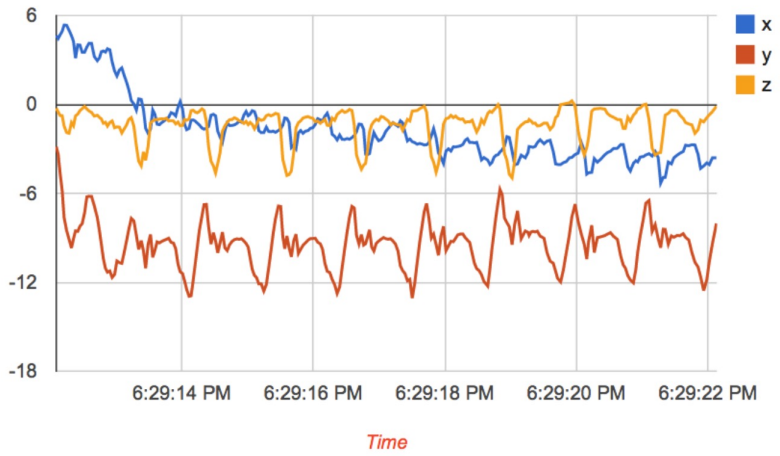
Orange Example

https://orange.biolab.si/

University *of Ljubljana*
Faculty *of Computer and Information Science*

Is Classification Always Necessary?

- Some applications allow simpler approaches
  - e.g. step counting
- Collect raw accelerometer data (noisy)



- Filtered the data

- Detect steps by calculating the intensity and detecting when the mean is crossed

Machine Learning on Android

University *of Ljubljana*
Faculty *of Computer and Information Science*

Machine Learning on Android

- On-device vs Cloud-based
  - Latency: c.b. depends on network speed
  - Energy: o.d. can deplete device resources
  - Privacy: o.d. preserves privacy
  - Model complexity: c.b. allows more complex models
- Cloud-based:
  - ML Kit, Google Cloud API
- On-device:
  - ML Toolkit – very simple models
  - WEKA ports – more complex traditional ML models
  - TensorFlow Lite and ML Kit – deep learning models

ML Toolkit

github.com/vpejovic/MachineLearningToolkit

- Simple on-device learning
  - Tree-based, Naïve Bayes, and Density Clustering
- Features:
  - Classifier persistence (as an internal file)
  - Loading external classifier (as a file)
- Usage - add as dependency:
  - 'si.uni_lj.fri.lrss.machinelearningtoolkit:mltoolkit:1.2’

ML Toolkit Example

University *of Ljubljana*
Faculty *of Computer and Information Science*

Weka Ports

- Weka
  - A large collection of machine learning algorithms implemented in Java
  - https://www.cs.waikato.ac.nz/ml/weka/index.html
- For Android:
  - No official port, but it's all Java anyways
  - Download .jar and include it in your project
- Using pretrained Weka models in Android:
  - A great lab by prof Campbell, Darmouth College: https://www.cs.dartmouth.edu/~campbell/cs65/lecture22/lecture22.html

ML Kit and TensorFlow Lite

- Firebase – a framework supporting mobile app development
  - Messaging, authentication, database, monitoring, etc.
- ML Kit – a part of Firebase dedicated to machine learning
  - On-device and cloud-based
  - Pretrained models for text recognition, face detection, object detection and tracking, barcode scanning, etc.
- TensorFlow – a library for NN programming
  - ML Kit's models use TensorFlow Lite under the hood
  - Create own models in TensorFlow and use in Andorid

TODO

- Machine Learning lab from Platform Based Development course materials on Ucilnica
- Read the following article about building a classifier:
  - Classification: Basic Concepts, Decision Trees, and Model Evaluation, Tan et al.
- Complete the second lab
- Read the deep learning article for next week