

Introduction to Pattern Recognition and Anomaly Detection

Good Morning!

- 1) Welcome to Day 2!
- 2) We will start at 9am sharp

Grab your coffee/team, sit back and relax for now 😊

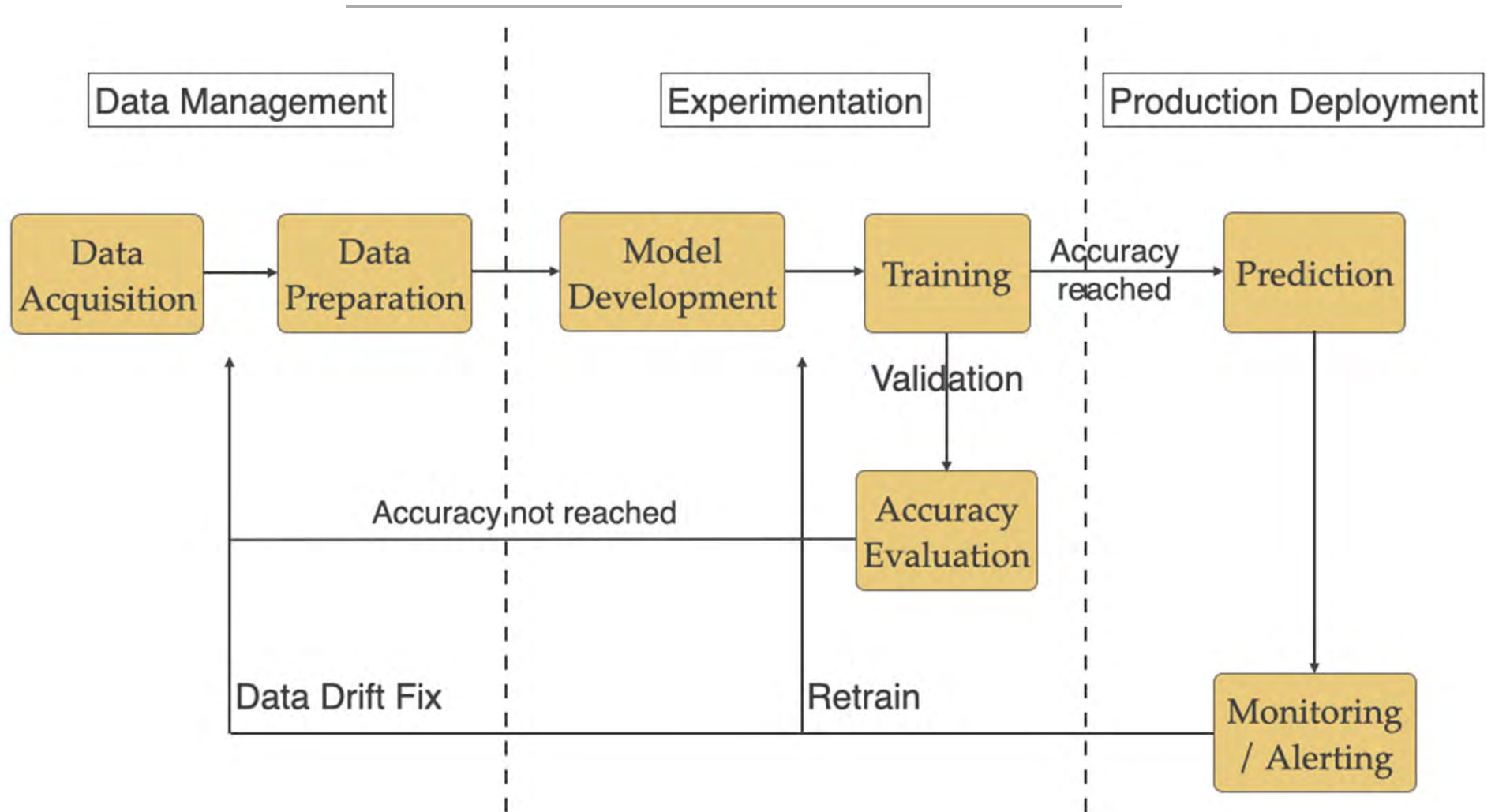


Programme

Day 1	<p>Introduction to Pattern Recognition and their techniques Activity – KNN & K-Means (iris)</p> <p>Common Python Lib - Numpy Activity - Numpy</p>	<p>Common Python Lib – Pandas, matplotlib, Scikit-learn Activity – Pandas, Matplotlib, Scikit-learn</p>
Day 2	<p>Data gathering to prepare for training and testing data Activity – Data Cleaning</p> <p>Use low-pass filter and simple moving average to detect abnormalities Activity – Simple Anomaly Detector</p>	<p>Introduction to anomaly detection and their techniques Introduction to H2O</p> <p>Activity - IRIS, MNIST, Fashion MNIST</p>
Day 3	<p>Activity – Credit card fraud with H2O/Isolation forest</p> <p>Activity – Intrusion detection system</p>	<p>Introduction to Alibi-Detect</p> <p>Activity – Intrusion detection system Activity – Image anomaly detection (MVTec)</p>



Machine Learning workflow





Data Preparation

- Data can be gathered from many sources and the quality of the data may not be 'clean'
- Data preparation involves the following
 - Data Cleaning
 - Segmentation of Data for Training, Validation and Testing
- Data scientist will be spending most of the time in data preparation



Similarity to data cleansing, we need to clean up the data as the situation when it is gathered may not ideal



Data Cleaning

- Data that are gathered are usually far from ideal as there may be:
 - Incorrect Data Format (e.g. Date should be ivy/mmm/did)
 - Missing Data
 - Incorrect Data Type (e.g. Data is numeric but a string value is given)
 - Data Duplication
 - Etc.



*It is a tedious
tasks that need to
be done*



WHY IS IT IMPORTANT!

- Poor data could lead to wrong classification
- When there is wrong classification of data, it will lead to wrong predication and outcome.
- Wrong predication and outcome will be 'costly' to the organization

For example, the Land Transport Authority may like to investigate the public transport usage. If the data is not clean, LTA may not have the wrong conclusion and prediction.



Data Cleaning

- In Data Science (e.g. Data Analytics, AI), it will require large amount of data and manually cleaning of these data will be impossible
- Tools like Python will be able to automate the process of cleaning the data based on 'rules'
- The data will be manipulated based the rules set in the programmed



Explanation to the Data Cleaning Programme

```
# Scenario 1
# This programme is to clean the data from the input file
import pandas as pd
import numpy as np

#Example
df = pd.read_csv('Data/dirty_data.csv')

#Create a duplicate for df
df_Ex1= df.copy()

# Rule 1
# This sets of routines will null all non-numeric values from the individual columns
df_Ex1['floor_size'] = pd.to_numeric(df_Ex1['floor_size'], errors="coerce")

# Scenario 1 Exercise 1: Add the codes required to null all non-numeric or blank values
# ..... COMPLETE CODE .....
#-----

# Computer the means
df_mean=df_Ex1.mean()

# Rule 2
# This set of routines are to convert null value to the mean value of the columns
df_Ex1.loc[df_Ex1['floor_size'].isnull(), 'floor_size'] = df_mean['floor_size']

# Scenario 1 Exercise 2: Add the codes required to convert all null values to the mean value
# ..... COMPLETE CODE .....
#-----

#Save to CSV file
df_Ex1.to_csv('Data/clean_data_scenario1.csv')
```

Creating a duplicate of the dataframe df

Changes made to df_Ex1 will not change the values of df1

Remember the loc function
df.loc(row, column)

This step will nullify ALL non-numeric or blank values for a particular column

Boolean function that will return the index of row that meet the condition

Save DataFrame to a csv file

Explanation to the Data Cleaning Programme

OFFICIAL (CLOSED) \ NON-SENSITIVE



```
# Scenario 2
# This programme is to clean the data from the input file
import pandas as pd
import numpy as np

#Example
df = pd.read_csv('Data/dirty_data.csv')

#Create a duplicate for dataframe df
df_Ex2 = df.copy()

df_Ex2a = df_Ex2.drop(df_Ex2[df_Ex2['median_income'].str.isalpha()== True].index)
#Complete the code
# Add the required codes for the rest of the columns
```

Drop the rows from the
DataFrame

Function that
check whether the
value is characters



Writing the Data Cleaning Programme

Scenario 1

- Rules for cleaning the data
 - Ensure that all non-numeric characters are nullified
 - Ensure that all nullified values are given the mean values of the respective columns

Scenario 2

- Rules for cleaning the data
 - Remove rows where there are non-numeric value
 - Ensure that all nullified values are given the mean values of the respective columns



Data Preparation for building the model



- Machine Learning will require a model to perform prediction
- The model used are usually statistical algorithms that will perform the 'prediction'
- Prediction are based on the probability of the data point that matches certain criterion

The last step in Data Preparation



- After the data are 'cleaned', it is important that the data split into the following datasets
 - Training
 - Validation
 - Test

Note in some case. validation is known as the test and test is known as the holding datasets

- It is important to note that the data are randomly split into the respective datasets (Training, Validation and Test)

Purpose of the Respective Datasets



- Training
 - Training datasets are used for training the models. For each data that passed through the model, it will 'refine' the model.
- Validation
 - Validation datasets are used for the checking the model.
- Testing
 - Test datasets plays the similar role as the validation datasets but it is to ensure that the model best fit the hypothesis

Normal Ratio of splitting the data:

Training :70% Validation 20% Testing : 10%

Note this ratio is able to change based on requirements



Example of Data Splitting Programme

- Ratio of Training : Validation: Test is 70:20:10

```
df = pd.read_csv('../Data/out/clean_data_scenario1.csv')  
  
#Create a duplicate for dataframe df  
df_Example= df.copy()  
  
train, validate, test = np.split(df_Example.sample(frac=1), [int(.7*len(df_Example)), int(.9*len(df_Example))])
```

Split location in the array



Activity 6

- 3 cleaning exercises
 - non-numeric characters are set to null
 - non-numeric characters are dropped
 - Splitting the data for training and testing.



Step 1:

Watch and listen to the instructor's demonstration



15 mins

Step 2:

Work through the activities



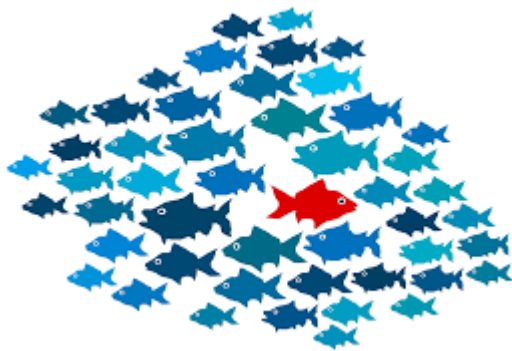
30 mins

Individual Activity

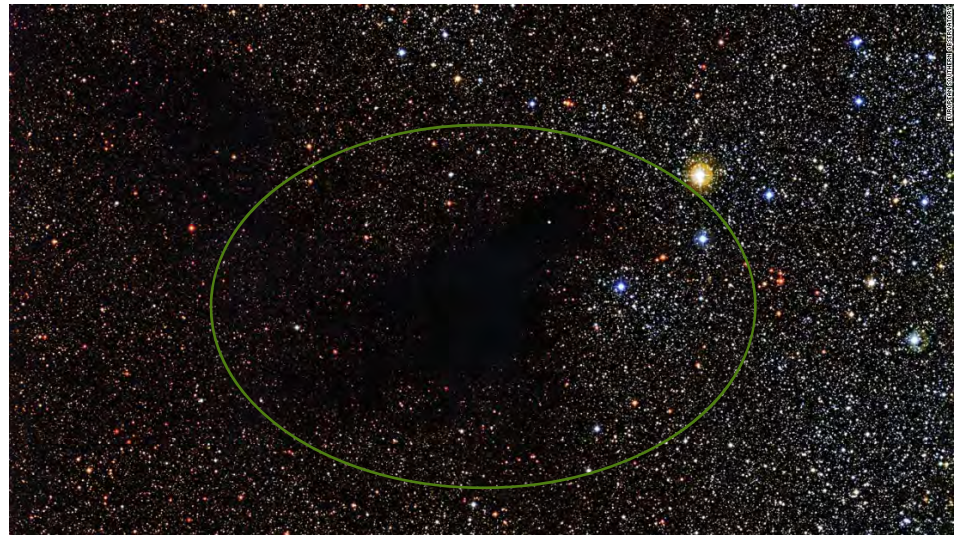


Anomalies

- What are anomalies?
 - It is a series of events that does not conform to the expected patterns or outcomes



Where is the anomaly in this picture?



Machine Learning Algorithm can be used to detect such anomalies



Type of Anomalies

- Point Anomalies
 - A single instance of data that is too far from the rest
- Contextual Anomalies
 - Anomalies are more context specific and is common in time series data
- Collective Anomalies
 - A set of data instances collectively that collectively determines anomalies (e.g. High processing time + High Network traffic = Potential Cyber attack)

Machine Learning Anomalies Detection Techniques



- Density-Based Anomaly Detection
 - It is based on the K-nearest neighbors algorithm
 - It is based on assumption that the normal data points occur near a dense neighborhood. Any data points are far away are consider as anomalies.
- Clustered-Based Anomaly Detection
 - Based on the k-means algorithms to detect anomalies in unsupervised learning
 - Data points that belong to a clusters are determined by the distance from the centroids of the cluster. Data points are outside these groups are considered as anomalies

Machine Learning Anomalies Detection Techniques



- **Moving Average (Low Pass filter) using Discrete Linear Convolution**
 - Convolution is a mathematical operation that is performed on two function to produce the third function
- It is uses a time-series to analyze the data and detect the anomalies in the data
- Moving average extract trends of a time series
- In the presence of outlier, it will affect the moving averages
- It will attempt to filter high value data and allow only low values to pass through

Disadvantage of Moving Average (Low-pass filter)



- There are many ways in which an anomaly occurring in a time series may be defined.
- For detecting anomalous sub-sequences, the exact length of the subsequence is often unknown.
- Performances of many anomaly detection algorithms are highly susceptible to noise in the time series data, since differentiating anomalies from noise is a challenging task.
- Time series in real applications are usually long and as the length increases the computational complexity also increases.

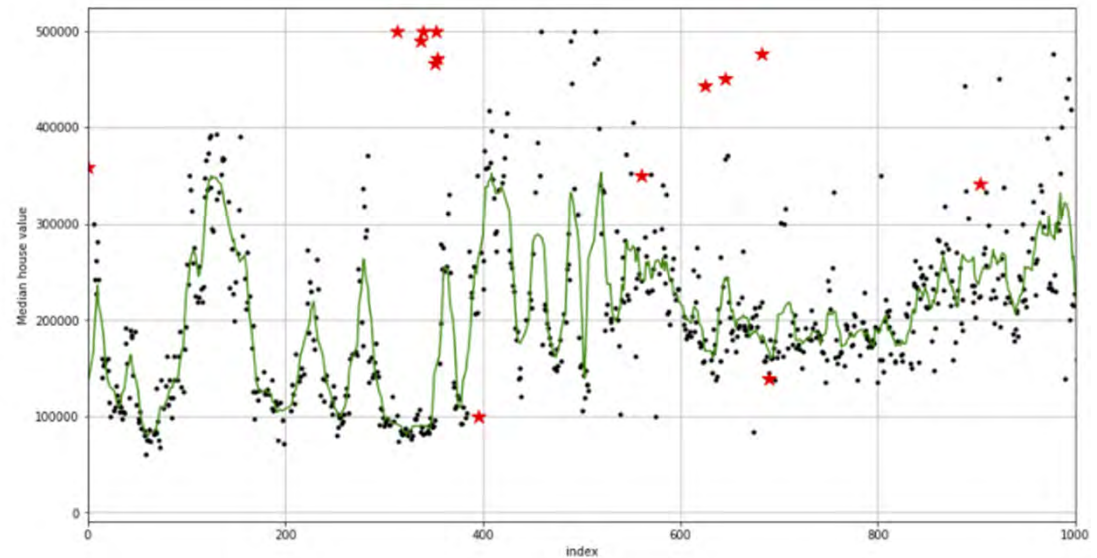
Pre-briefing for writing Low Pass filter Programme



- The following are key functions in the programme:
 - moving_average
 - explain_anomalies
 - explain_anomalies_rolling_std
 - plot_results
- You need not make changes to the function
- The partial code given will explain what need to be done



Activity 7 - Low pass filter detector



Exercise

Develop a low pass filter detector using the sunspot dataset (sunspot.csv)

Step 1:

Watch and listen to the instructor's demonstration



20 mins

Step 2:

- Do on your own



30 mins

Optional Activity



H2O

- What is H2O?
 - Open-source, rich analytics platform that provides both machine learning algorithms and high-performance parallel computing abstractions.
 - Built around a Java Virtual Machine optimized for in-memory processing of distributed data collections.
 - Platform can be used via web-based UI or via languages such as Python, R, Java, Scala, and JSON in a REST API.
 - After loading into H2O, data is represented in an H2O frame.



Advantages of H2O

- Can run locally or deployed in a cluster on top of Spark or Hadoop MapReduce.
 - Scalability
- Why we choose H2O?
 - It provides a built-in class, H2OAutoEncoderEstimator, that we use for unsupervised learning.
 - Ease of use
 - Well-suited for curious users who want to learn and experiment
 - Reliability
 - Good fit for enterprise environments that care about production aspects



Installation

- Pre-requisites
 - Python 2.7 or Python 3.5
 - pip, Python's package manager
 - Java platform (JDK) 9
- Test H2O

```
python  
import h2o  
h2o.init()
```

```
Connecting to H2O server at http://127.0.0.1:54321... successful.  
-----  
H2O cluster uptime:      05 secs  
H2O cluster version:    3.16.0.4  
H2O cluster version age: 10 hours and 20 minutes  
H2O cluster name:       H2O_from_python_student_q8gvwl  
H2O cluster total nodes: 1  
H2O cluster free memory: 477.1 Mb  
H2O cluster total cores: 1  
H2O cluster allowed cores: 1  
H2O cluster status:     accepting new members, healthy  
H2O connection url:     http://127.0.0.1:54321  
H2O connection proxy:     
H2O internal security:   False  
H2O API Extensions:     AutoML, Algos, Core V3, Core V4  
Python version:         2.7.6 final  
-----  
>>> 
```



Installation



▼ Setup and init H2O

```
[ ] ! apt-get install default-jre  
    ! java -version  
    ! pip install h2o  
    import h2o
```



H2O GUI (Flow)

The screenshot displays the H2O Flow web interface. At the top, the browser address bar shows '127.0.0.1:54321/flow/index.html'. The main header includes the 'H2O FLOW' logo and a navigation menu with options: Flow, Cell, Data, Model, Score, Admin, and Help. Below the header, the title 'Untitled Flow' is visible. A toolbar with various icons for file operations and execution is present. The main workspace area shows a command prompt with the text 'assist' and a timestamp of '379ms'. On the left side, an 'Assistance' panel is open, listing various routines and their descriptions.

Routine	Description
importFiles	Import file(s) into H ₂ O
getFrames	Get a list of frames in H ₂ O
splitFrame	Split a frame into two or more frames
mergeFrames	Merge two frames into one
getModels	Get a list of models in H ₂ O
getGrids	Get a list of grid search results in H ₂ O
getPredictions	Get a list of predictions in H ₂ O
getJobs	Get a list of jobs running in H ₂ O
buildModel	Build a model
runAutoML	Automatically train and tune many models
importModel	Import a saved model
predict	Make a prediction



Getting Started

- Import and initialise module
 - **h2o.init** : Connect to a running H2O instance using all CPUs on the host.
- Data preparation
 - **h2o.importFile**: Import a file into H2O from a server-side path, and parse it.
 - **h2o.splitFrame**: Split an existing H2O dataset according to user-specified ratios
 - **.insert_missing_values()** – Replaces a user-specified fraction of entries in an H2O dataset with missing values.
- Define and train model
 - **H2ORandomForestEstimator()**, **H2ODeepLearningEstimator()**
 - **.train** – train the model
- Evaluate model
 - Use **scikit-learn**, **score_history()**, **model_performance()**, **confusion_matrix()**
- Predict
 - **.predict()**
- Load & save model
 - **h2o.save_model()**, **h2o.load_model()**



Estimators

- Supervised
 - Cox Proportional Hazards
 - Neural Networks
 - Distributed Random Forest (DRF)
 - Generalised Linear Model (GLM)
 - Generalised Additive Models (GAM)
 - Gradient Boosting Machine (GBM)
 - Naïve Bayes Classifier
 - RuleFit
 - Stacked Ensembles
 - Support Vector Machine (SVM)
 - XGBoost
- Unsupervised
 - Aggregator
 - Generalised Low Rank Models (GLRM)
 - Isolation Forest
 - K-Means Clustering
 - Principal Component Analysis



Supervised Algorithms

- **Cox Proportional Hazards**
 - most widely used approach for modeling time to event data
- **Deep Learning (Neural Networks)**
 - based on a multi-layer [feedforward artificial neural network](#) that is trained with stochastic gradient descent using back-propagation.
- **Distributed Random Forest(DRF)**
 - powerful classification and regression tool. When given a set of data, DRF generates a forest of classification or regression trees, rather than a single classification or regression tree. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/drf.html>)
- **Generalised Linear Model (GLM)**
 - estimate regression models for outcomes following exponential distributions. Includes Gaussian, Poisson and Binomial regression (classification) etc. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/glm.html>)
- **Generalized Additive Models (GAM)**
 - a GLM in which the linear predictor depends linearly on predictor variables and smooth functions of predictor variables.



Supervised Algorithms

- **Gradient Boosting Machine (GBM)**
 - a forward learning ensemble method. The guiding heuristic is that good predictive results can be obtained through increasingly refined approximations (for Regression and Classification)
(<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/gbm.html>)
- **Naïve Bayes Classifier**
 - a classification algorithm that relies on strong assumptions of the independence of covariates in applying Bayes Theorem. The Naïve Bayes classifier assumes independence between predictor variables conditional on the response, and a Gaussian distribution of numeric predictors with mean and standard deviation computed from the training dataset.
- **RuleFit**
 - combines tree ensembles and linear models to take advantage of both methods: the accuracy of a tree ensemble and the interpretability of a linear model. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/rulefit.html>)



Supervised Algorithms

- **Stacked Ensembles**
 - use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/stacked-ensembles.html>)
- **Support Vector Machine (SVM)**
 - A supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/svm.html>)
- **XGBoost**
 - A supervised learning algorithm that implements a process called boosting to yield accurate models. Boosting refers to the ensemble learning technique of building many models sequentially, with each new model attempting to correct for the deficiencies in the previous model. (<https://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/xgboost.html>)



Unsupervised Algorithms

- **Aggregator**
 - clustering-based method for reducing a numerical/categorical dataset into a dataset with fewer rows. (<http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/aggregator.html>)
- **Generalised Low Rank Models (GLRM)**
 - an algorithm for dimensionality reduction of a dataset.
- **Isolation Forest**
 - similar in principle to Random Forest and is built on the basis of decision trees. Isolation Forest, however, identifies anomalies or outliers rather than profiling normal data points. Isolation Forest isolates observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of that selected feature. This split depends on how long it takes to separate the points.



Unsupervised Algorithms

- **K-Means Clustering**
 - a form of unsupervised learning that tries to find structures in the data without using any labels or target values. Clustering partitions a set of observations into separate groupings such that an observation in a given group is more similar to another observation in the same group than to another observation in a different group.
- **Principal Component Analysis**
 - closely related to Principal Components Regression. The algorithm is carried out on a set of possibly collinear features and performs a transformation to produce a new set of uncorrelated features.
 - PCA is commonly used to model without regularization or perform dimensionality reduction. It can also be useful to carry out as a preprocessing step before distance-based algorithms such as K-Means since PCA guarantees that all dimensions of a manifold are orthogonal.



AutoML

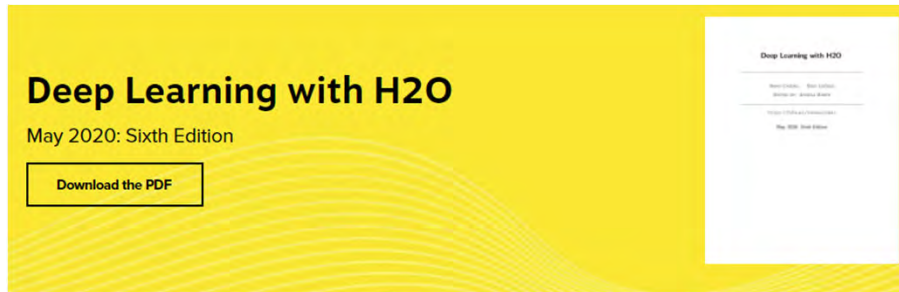
- easy-to-use interface which automates the process of training a large selection of candidate models.
- a simple wrapper function that performs a large number of modeling-related tasks that would typically require many lines of code, and by freeing up their time to focus on other aspects of the data science pipeline tasks such as data-preprocessing, feature engineering and model deployment.



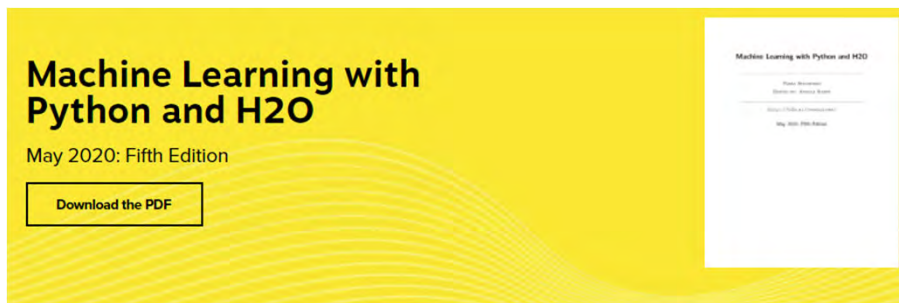


Resources

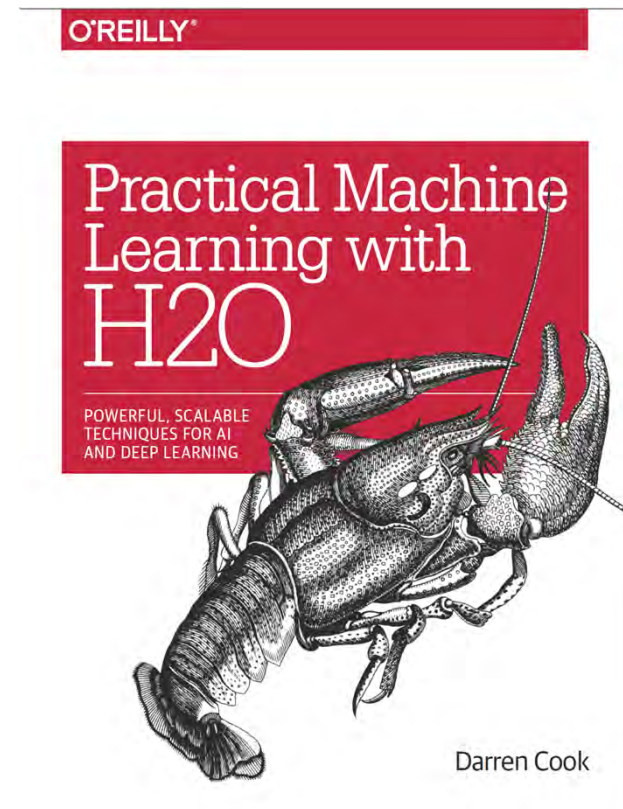
- H2OFrame Pandas DataFrame cheat sheet https://github.com/h2oai/h2o-3/blob/master/h2o-docs/src/cheatsheets/Python_H2OFrame_PandasDataFrame_Parity.md
- h2o cheat sheet : https://ugoproto.github.io/ugor_doc/pdf/h2o.pdf



<https://www.h2o.ai/resources/booklet/deep-learning-with-h2o/>



<https://www.h2o.ai/resources/booklet/deep-learning-with-h2o/>





Activity 8 – H2O IRIS

Samples
(instances, observations)

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...
50	6.4	3.5	4.5	1.2	Versicolor
...
150	5.9	3.0	5.0	1.8	Virginica

Features
(attributes, measurements, dimensions)

Class labels
(targets)

Petal

Sepal

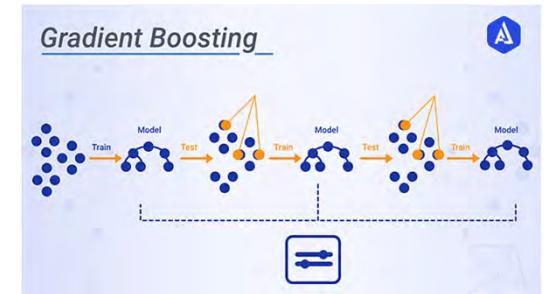
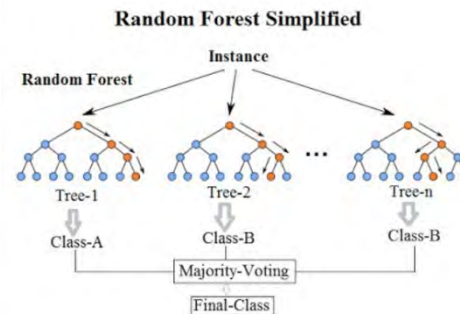


Image credits: [Principal Component Analysis](#) by [Sebastian Raschka](#)



Exercise

Use another supervised learning algorithm from H2O documentation to perform the classification

<http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science.html#supervised>

Step 1:

Watch and listen to the instructor's demonstration



15 mins

Step 2:

- Do on your own



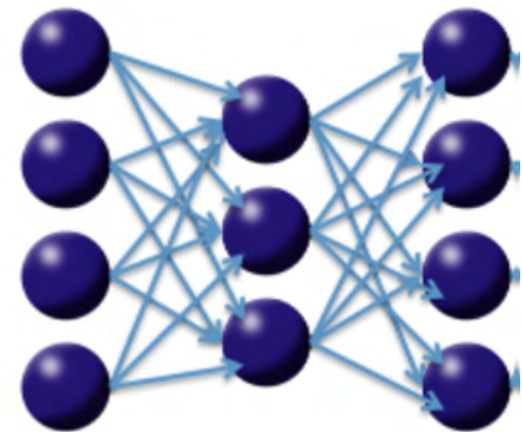
30 mins³⁷

Individual Activity



Autoencoders

- Approach AD using a semi-supervised learning
 - Identify a set of data that represents the normal distribution. Non anomaly.
 - Learn what “normal” means from this training dataset. The trained model will provide a sort of metric in its mathematical definition; i.e. a function mapping every point to a real number representing the distance from another point representing the normal distribution.
- By selecting the right threshold, we can achieve the desired trade-off between precision (fewer false alarms) and recall (fewer missed detections)
- Robust to noise
- Uses MSE as the loss function
- Reconstruct based on intermediate representations that minimize the training error. We learn those compression functions from the training set so that a normal pint is very likely to be reconstructed correctly, but **an outlier would have a higher reconstruction error** (the mean squared error between the original point and the reconstructed one)





auto-encoders to identify anomalies

- Dataset: MNIST digit anomaly recognition
- We are **NOT** predicting which number each image represents BUT **identifying badly written digit images**.
- We are doing **unsupervised learning** as we are discarding the response column containing the label (the digit).



Lab worksheet

- **Step 1: Import data from H2O repository**

```
train_with_label = h2o.import_file("http://h2o-public-test-  
data.s3.amazonaws.com/bigdata/laptop/mnist/train.csv.gz")  
test_with_label = h2o.import_file("http://h2o-public-test-  
data.s3.amazonaws.com/bigdata/laptop/mnist/test.csv.gz")
```

- **Step 2: Partition into train and test**

```
predictors = list(range(0,784))  
train = train_with_label[predictors]  
test = test_with_label[predictors]
```



- **Step 3: Create auto-encoder model**

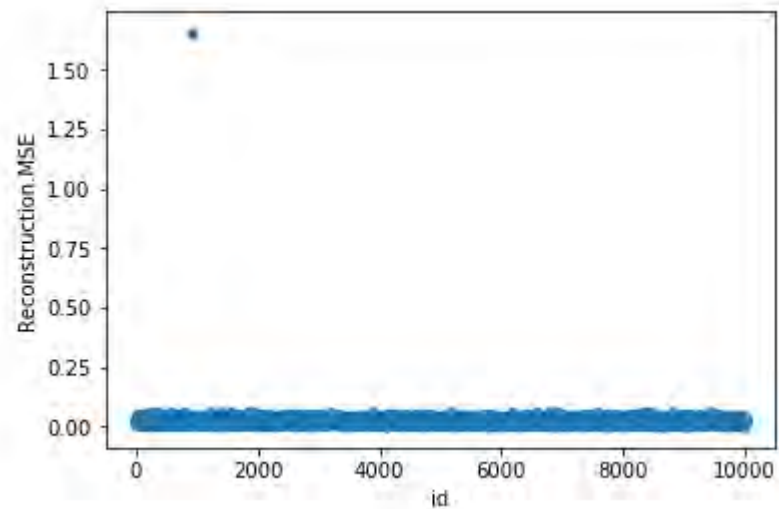
```
from h2o.estimators.deeplearning import H2OAutoEncoderEstimator  
model = H2OAutoEncoderEstimator(activation="Tanh", hidden=[20],  
ignore_const_cols=False, epochs=1)  
model.train(x=predictors, training_frame=train)
```

- **Step 4: Calculate reconstruction error**

```
test_rec_error = model.anomaly(test)
```

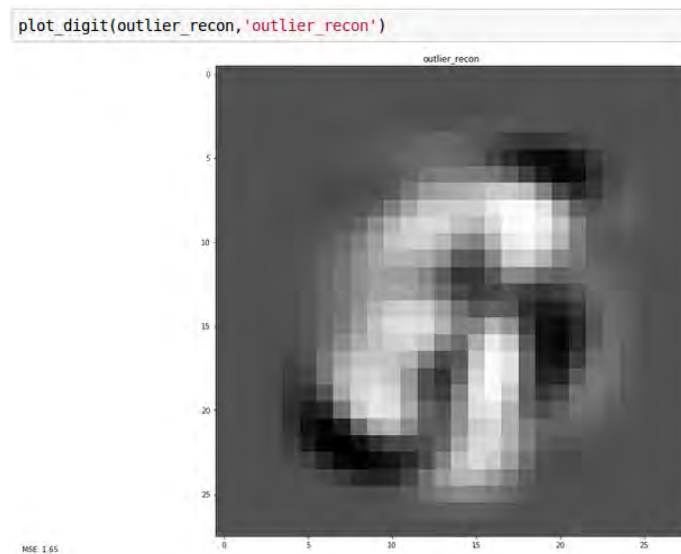
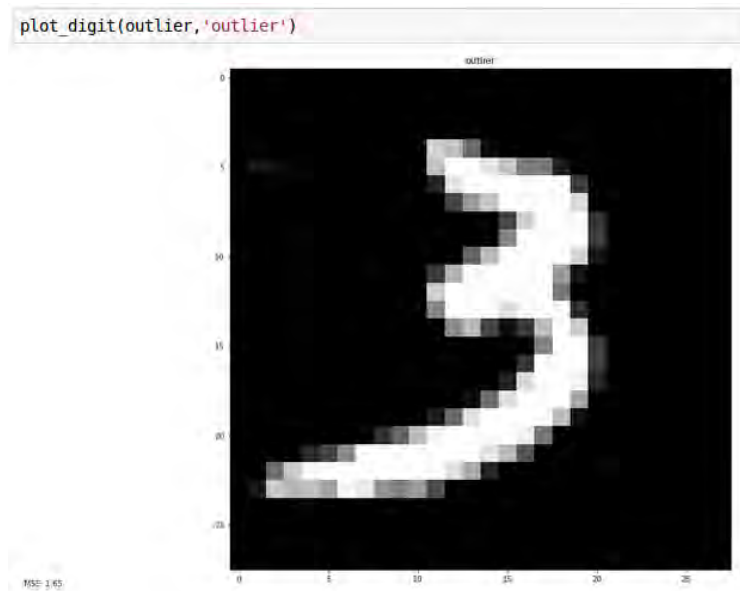


-
- Step 5: Create scatter plot of reconstruction error





- Step 6: Plot original outlier and its reconstructed version

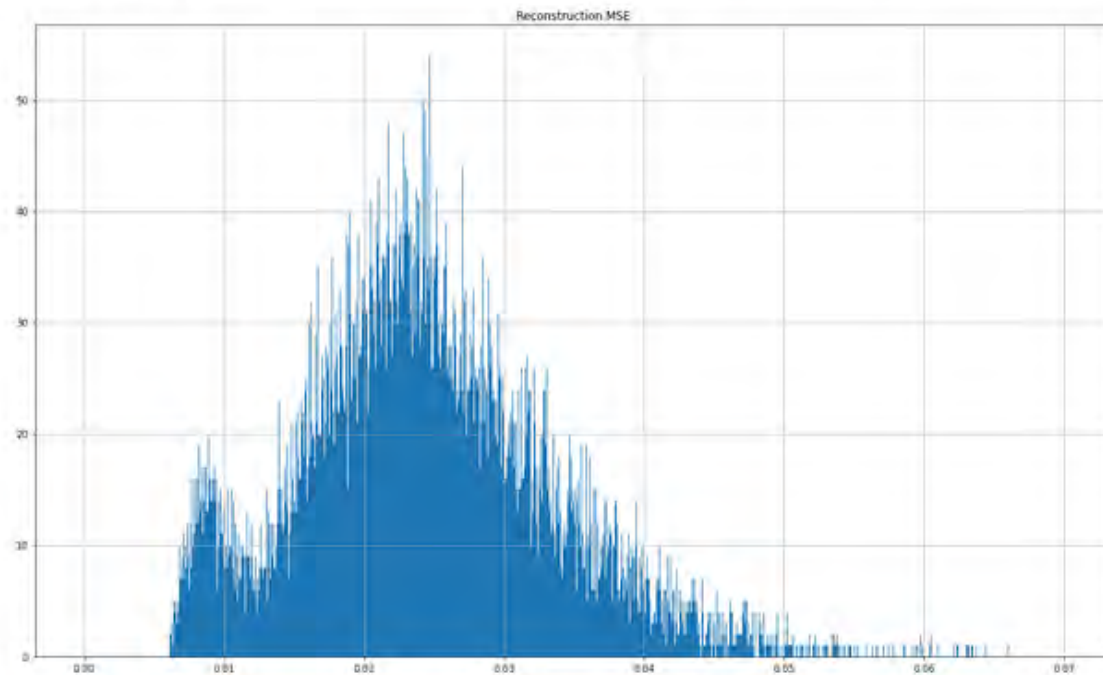




- Step 7: Plot error distribution of the remaining points

```
In [133]: test_rec_error.as_data_frame().hist(bins=1000, range=[0.0, 0.07])
```

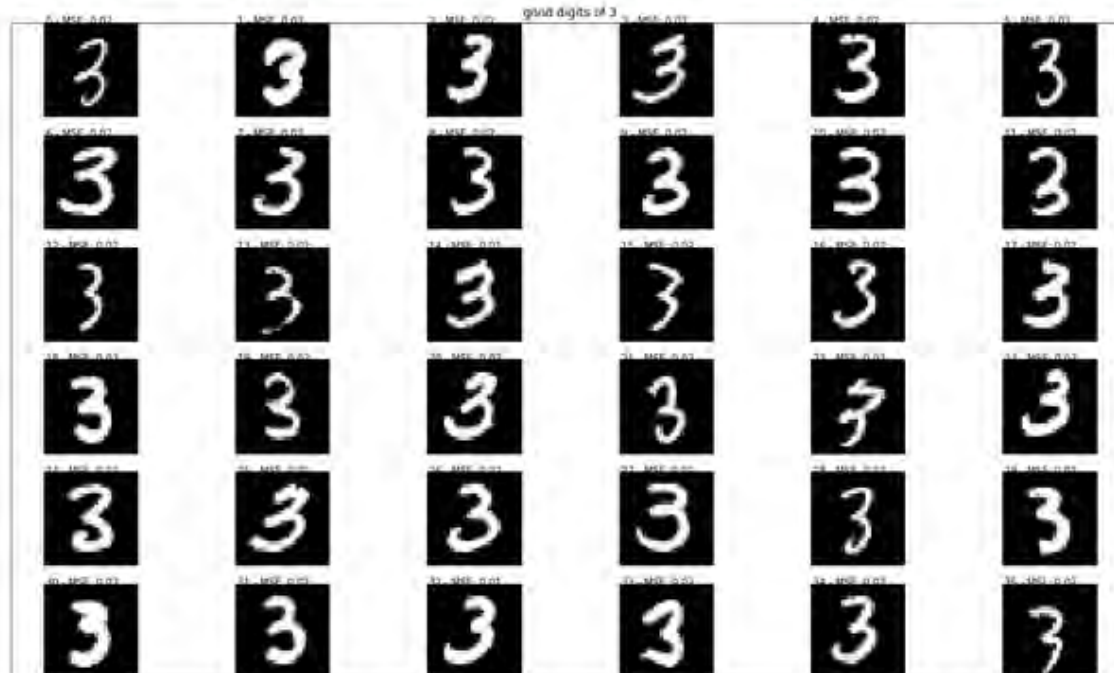
```
Out[133]: array([[<matplotlib.axes._subplots.AxesSubplot object at 0xada78eec>]],  
              dtype=object)
```





- Step 8: Visualize good digits of number three

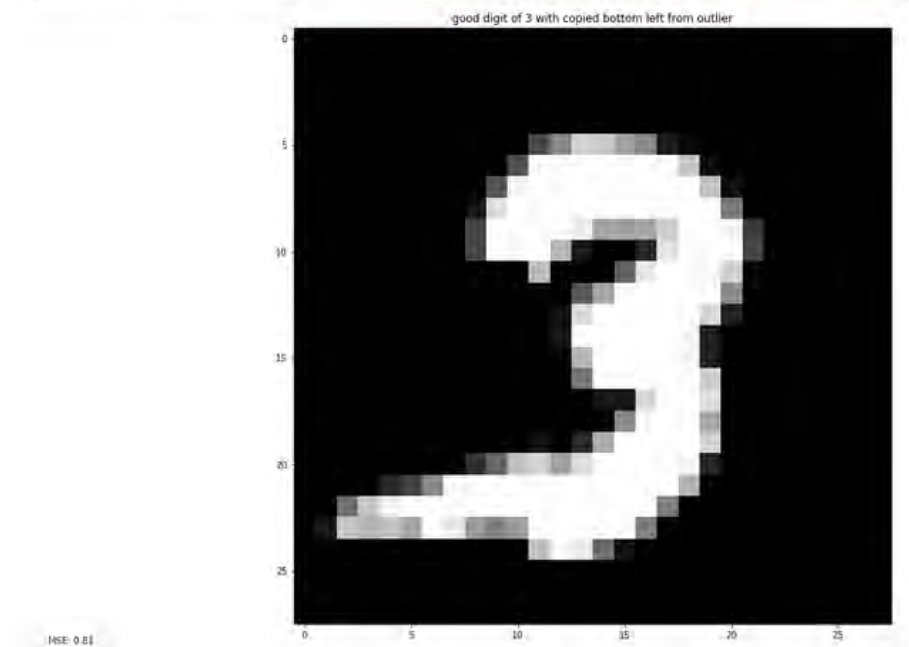
```
In [204]: plot_multi_digits(digits_of_3, 6, 6, "good digits of 3")  
#plot_multi_digits(model.predict(digits_of_3[predictors]).cbind(digits_of_3[
```





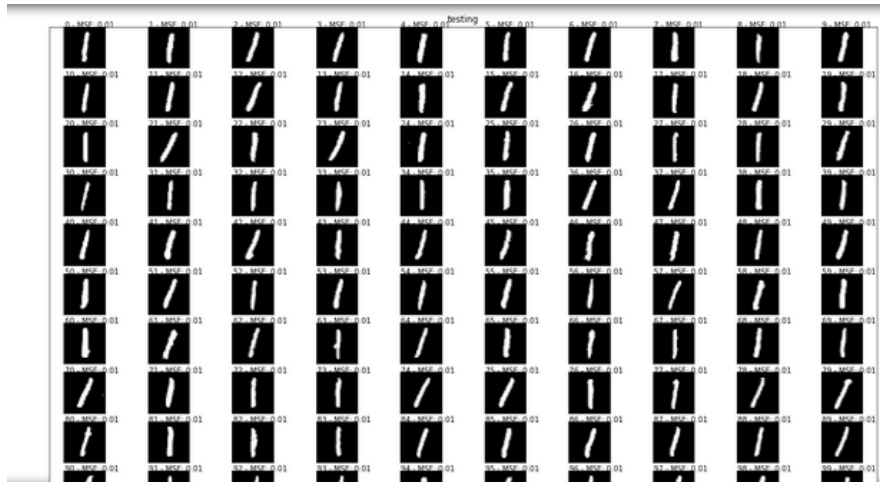
- Step 9: Visualize reconstructed version of the good digits of number three

```
In [196]: plot_digit(good_digit_of_3, 'good digit of 3 with copied bottom left from ou
```





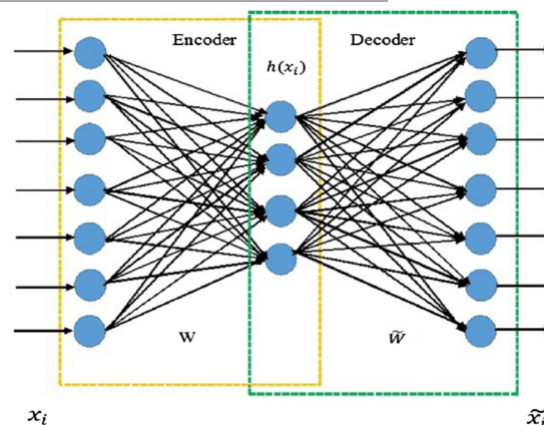
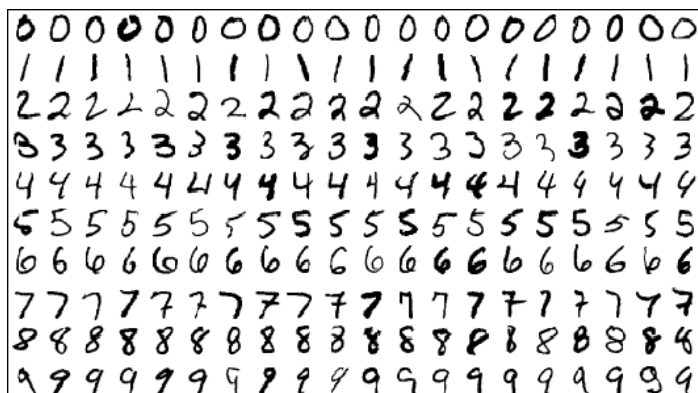
- Step 10: Manual inspection of bottom 100 (good) digits and 100 (ugly) digits



Viola! Indeed, the ugly digits represent the anomalies.



Activity 10 – H2O MNIST



Exercise

Fashion MNIST



Fashion-MNIST is a dataset of Zalando's article images—consisting of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. Fashion-MNIST to serve as a direct **drop-in replacement for the original MNIST** dataset for benchmarking machine learning algorithms. It shares the same image size and structure of training and testing splits.

* Use `fashion-mnist_test.csv` and `fashion-mnist-train.csv` provided

Step 1:

Watch and listen to the instructor's demonstration



20 mins

Step 2:

- Do on your own



30 mins

Optional Activity

OFFICIAL (CLOSED) \ NON-SENSITIVE

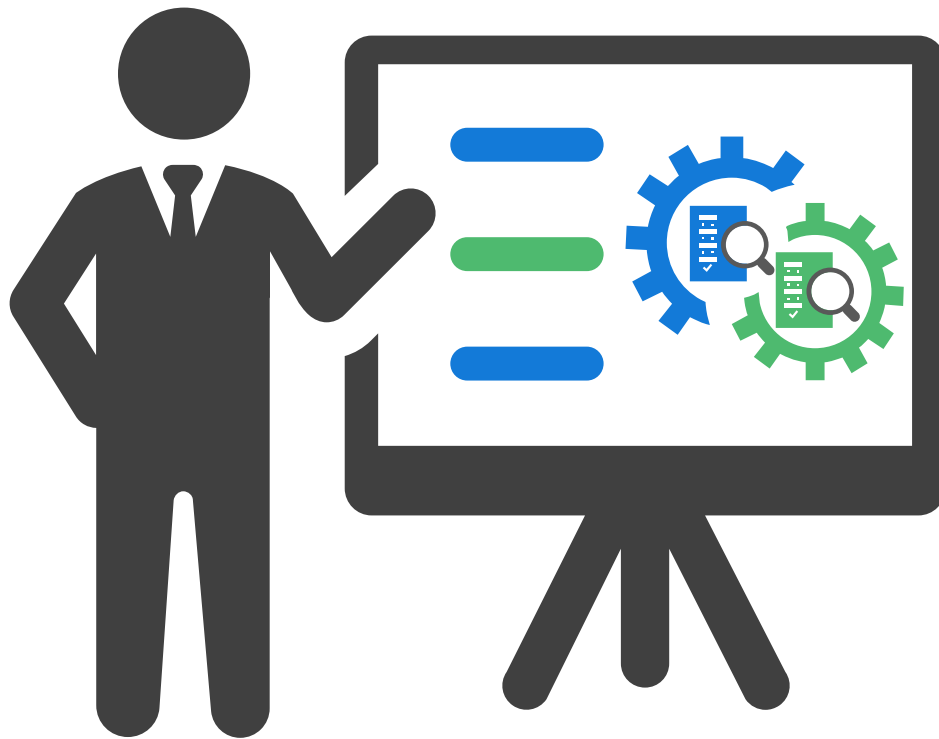


Quiz

https://bit.ly/kw_poll



SCAN ME



Summary

- Data gathering to prepare for training and testing data
- Use low-pass filter and simple moving average to detect abnormalities
- Introduction to anomaly detection and their techniques
- Introduction to H2O
- Supervised and unsupervised learning with Iris, MNIST, Fashion MNIST

Email

seow_khee_wei@rp.edu.sg

Telegram

@kwseow

Source code:

50