Cloud Computing & Big Data

PARALLEL & SCALABLE MACHINE LEARNING & DEEP LEARNING

Prof. Dr. – Ing. Morris Riedel
Associated Professor
School of Engineering and Natural Sciences, University of Iceland, Reykjavik, Iceland
Research Group Leader, Juelich Supercomputing Centre, Forschungszentrum Juelich, Germany

@Morris Riedel @MorrisRiedel @MorrisRiedel

PRACTICAL LECTURE 3.1

Building Scalable Machine Learning Pipelines with Apache Spark

September 24, 2020
Online Lecture

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

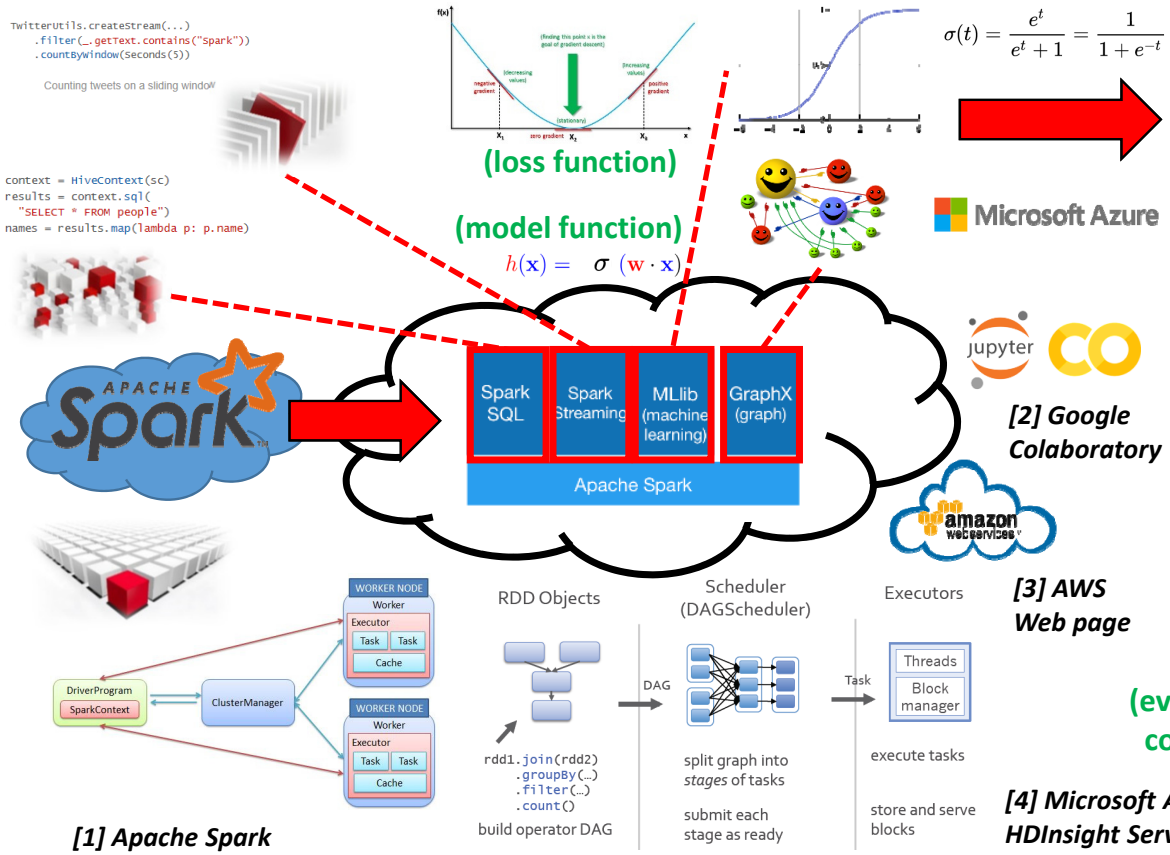
JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

DEEP Projects

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

Review of Lecture 3 – Apache Spark for Cloud Applications

- Apache Spark & Powerful Libraries



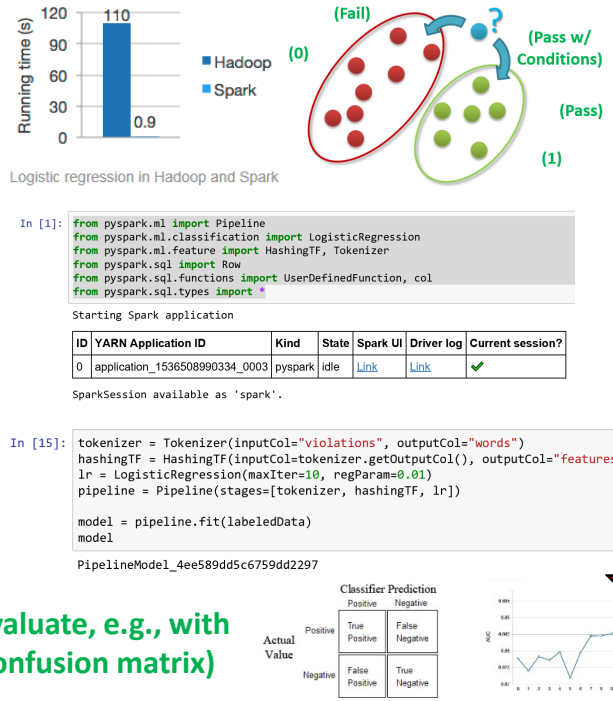
[1] Apache Spark

[2] Google Colaboratory

[3] AWS Web page

[4] Microsoft Azure HDInsight Service

- Machine Learning with Logistic Regression
  - Classification example does 'iterative learning'



(iteratively solving an optimization problem)

(evaluate, e.g., with confusion matrix)

[5] www.big-data.tips, 'gradient descent'

[6] Understanding Parallelization of Machine Learning Algorithms in Apache Spark

Outline of the Course

1. Cloud Computing & Big Data Introduction
2. Machine Learning Models in Clouds
3. Apache Spark for Cloud Applications
4. Virtualization & Data Center Design
5. Map-Reduce Computing Paradigm
6. Deep Learning driven by Big Data
7. Deep Learning Applications in Clouds
8. Infrastructure-As-A-Service (IAAS)
9. Platform-As-A-Service (PAAS)
10. Software-As-A-Service (SAAS)
11. Big Data Analytics & Cloud Data Mining
12. Docker & Container Management
13. OpenStack Cloud Operating System
14. Online Social Networking & Graph Databases
15. Big Data Streaming Tools & Applications
16. Epilogue

+ additional practical lectures & Webinars for our hands-on assignments in context

- Practical Topics
- Theoretical / Conceptual Topics

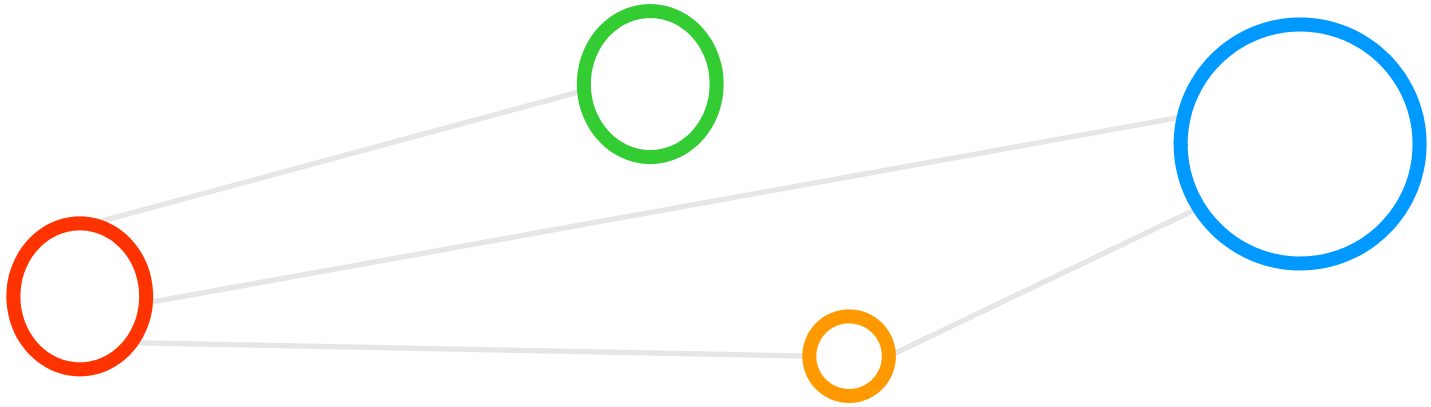
Outline

- Data Exploration via Cloud Demo in Supervised Learning
  - Formalization of Machine Learning Process in Diagram
  - Practical Step-wise Approach via MS Azure HDInsight
  - Binary Classification Application Example of Food Inspection
  - Understanding Mathematical Building blocks of Machine Learning
  - Necessity of Data Exploration & Relevance of Labelled Datasets

- **Promises from previous lecture(s):**
- ***Lecture 3:*** **Practical Lecture 3.1 offers a more practical introduction in a variety of machine learning algorithms using Apache Spark Mllib in Clouds**
- ***Lecture 3:*** **Practical Lecture 3.1 offers a practical introduction into optimization and its relationship to techniques like stochastic gradient descent**

- Machine learning via Cloud Demo using Apache Spark Pipelines
  - Continue the Formalization of Machine Learning Process in Diagram
  - Connect Machine Learning Steps with PySpark Approach in Clouds
  - Understanding Hypothesis Set & Learning Process via Optimization
  - Connecting Apache Spark PySpark Pipelines to Supervised Learning
  - Using Spark Libraries for Evaluation Options & Results Visualization

Data Exploration via Cloud Demo in Supervised Learning



Cloud Computing using Step-wise Approach via MS Azure HDInsight

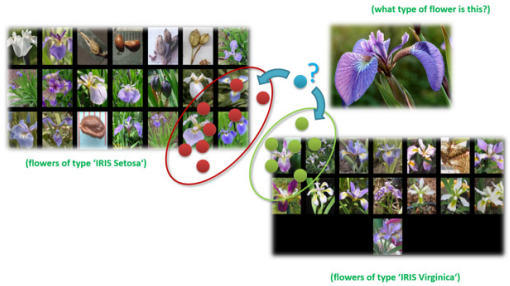
- Big Data Challenges
  - Jupyter, Anaconda and scikit-learn are great – but only for small data sets
  - Issues: Laptop too slow runtime (less CPUs/GPUs) and/or errors due to memory problems/limits

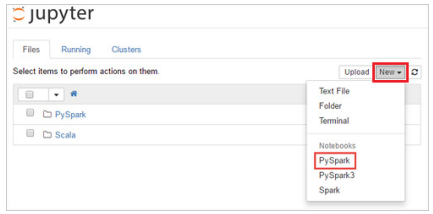
- Cloud Computing Step-wise Approach
  1. Check and/or create subscription (setup pay-per-use of resources, free in course)
  2. Deploy a Spark Cluster in HDInsight (prepare computing infrastructure) using a pre-configured Azure Resource Manager (ARM) template
  3. Startup Jupyter notebook using a PySpark kernel to use Apache Spark
  4. Create Spark session & check application dataset
  5. Initial data analysis, visualization & modeling
  6. Perform machine learning model on Spark Cluster
  7. Machine learning model evaluation & refinement

Microsoft Azure HDInsight

[4] Microsoft Azure HDInsight Service

[8] Jupyter

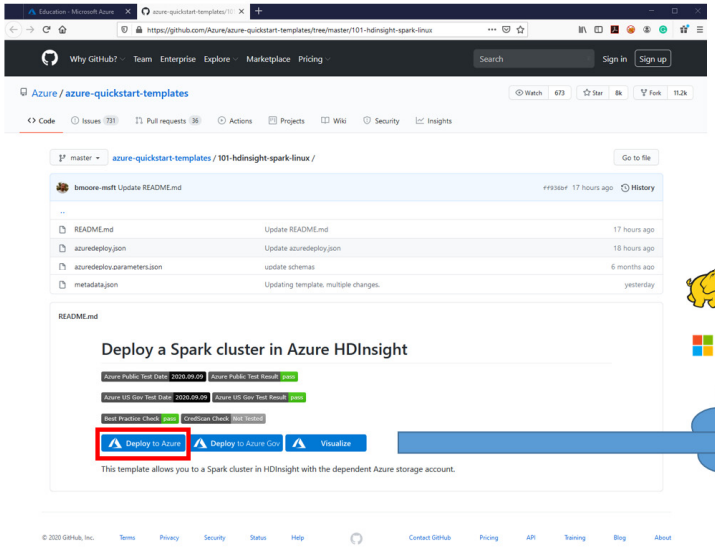


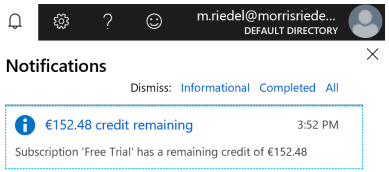


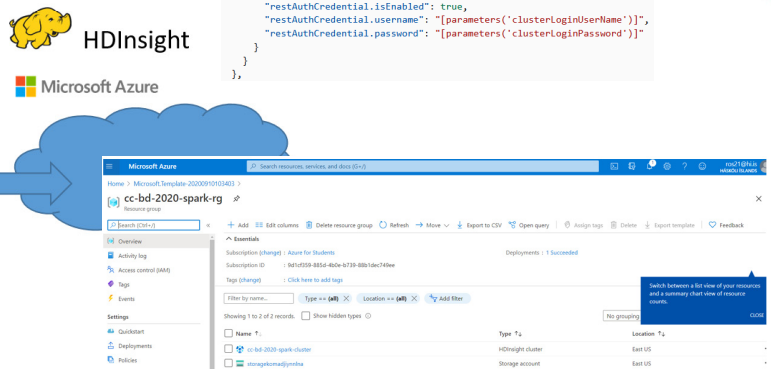
[7] Image sources: Species Iris Group of North America Database, www.signa.org

Step 1 & Step 2 Performed using Template & Possibility to Adapt & Automate

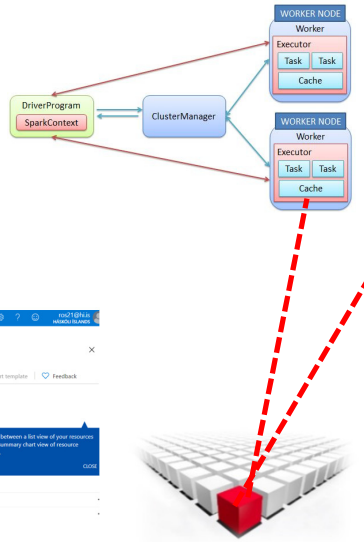
- Step 1: Check and/or Create Subscription
- Step 2: Deploy Spark Cluster in MS Azure HDInsight








```
"properties": {
  "clusterVersion": "3.6",
  "osType": "Linux",
  "tier": "Standard",
  "clusterDefinition": {
    "kind": "spark",
    "configurations": {
      "gateway": {
        "restAuthCredential.isEnabled": true,
        "restAuthCredential.username": "[parameters('clusterLoginUserName')]",
        "restAuthCredential.password": "[parameters('clusterLoginPassword')]"
      }
    }
  },
```



Review the template

The template used in this quickstart is from Azure Quickstart Templates.

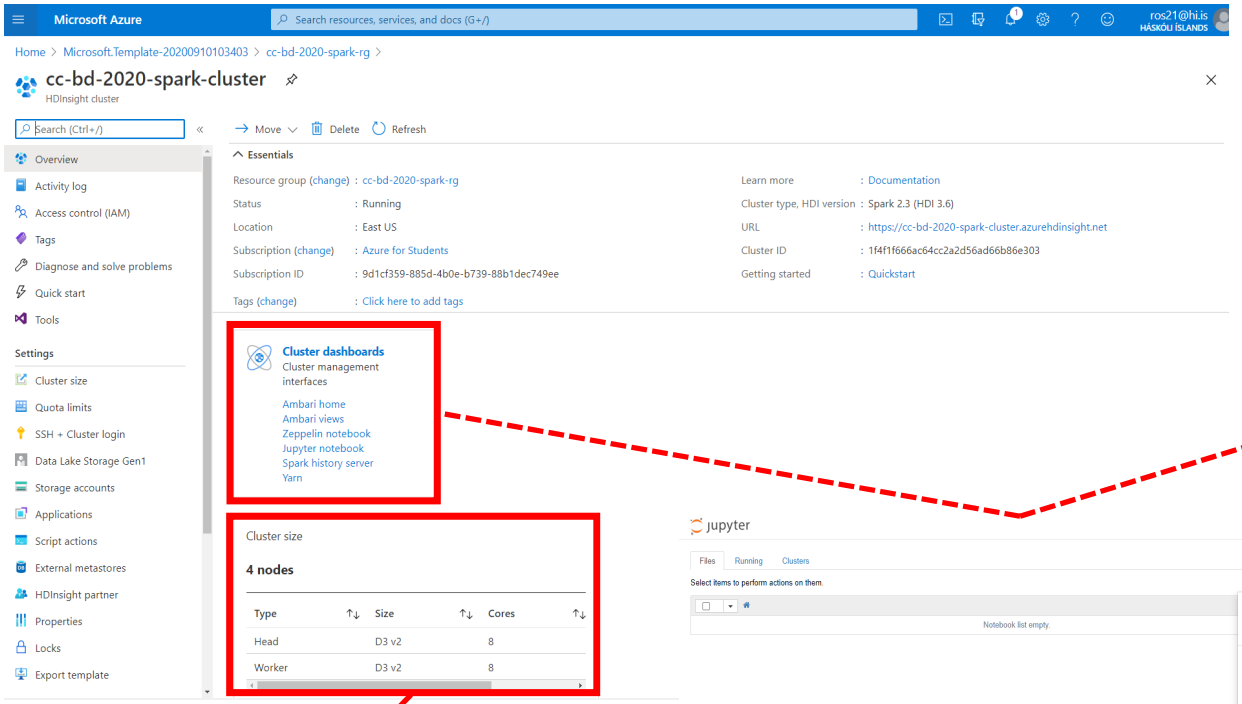
JSON

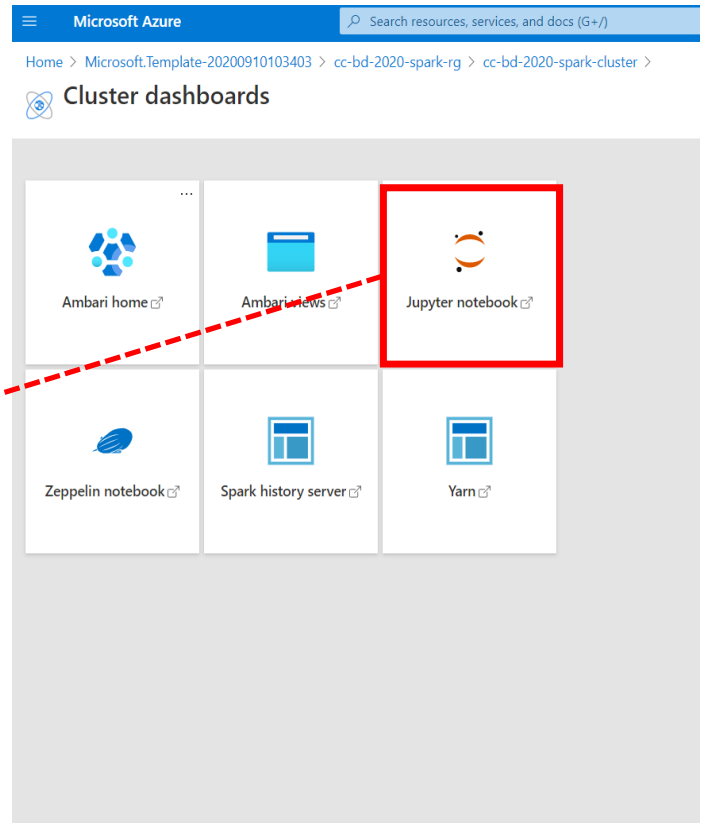
```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "clusterName": {
      "type": "string",
      "metadata": {
        "description": "The name of the HDInsight cluster to create."
      }
    },
    "clusterLoginUserName": {
      "type": "string",
      "metadata": {
        "description": "These credentials can be used to submit jobs to the cluster and to log into cluster dashboards."
      }
    },
```

```
"computeProfile": {
  "roles": [
    {
      "name": "headnode",
      "targetInstanceCount": 2,
      "hardwareProfile": {
        "vmSize": "[parameters('HeadNodeVirtualMachineSize')]"
      },
      "osProfile": {
        "linuxOperatingSystemProfile": {
          "username": "[parameters('sshUserName')]",
          "password": "[parameters('sshPassword')]"
        }
      }
    },
    {
      "name": "workernode",
      "targetInstanceCount": 2,
      "hardwareProfile": {
        "vmSize": "[parameters('WorkerNodeVirtualMachineSize')]"
      },
      "osProfile": {
        "linuxOperatingSystemProfile": {
          "username": "[parameters('sshUserName')]",
          "password": "[parameters('sshPassword')]"
        }
      }
    }
  ]
}
```

[9] Azure Portal Hub [11] MS Azure Storage

Step 3: Startup Jupyter Notebook & PySpark Kernel





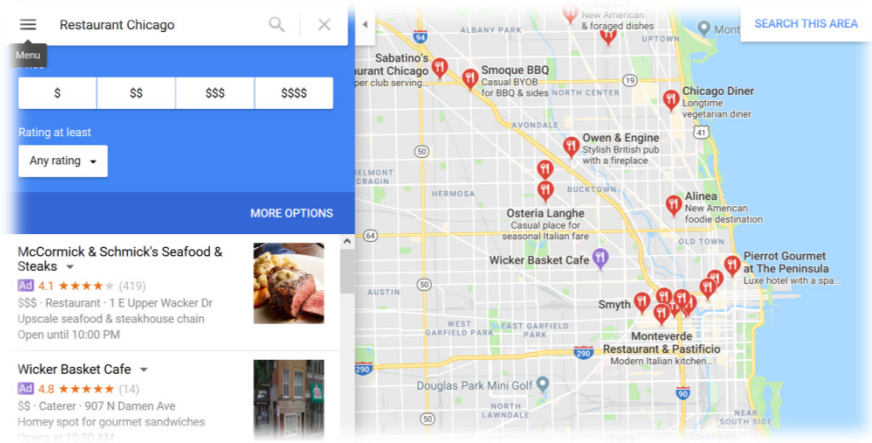
(number of nodes in a Spark cluster)

[1] Apache Spark

Food Inspection in Chicago: Advanced Application Example

1. Some pattern exists:
   - Believe in a pattern with ‘quality violations in checking restaurants‘ will somehow influence if food inspection pass or fail (binary classification)
2. No exact mathematical formula
   - To the best of our knowledge there is no precise formula for this problem
3. Data exists → Looks like Binary Classification Model is needed
   - Data collection from City of Chicago
   - Data is labelled & Outcome Pass/Fail wanted

- **The goal of the advanced machine learning application with food inspection of restaurants in the City of Chicago is to predict the outcome of food inspection of new Chicago restaurants given some of existing violations of older restaurants already obtained in Chicago**
- **A key question is if the new restaurant will pass or fail the inspection just based on violations**



Feasibility of Learning from Data – Formalization

- Theoretical framework underlying practical learning algorithms
  - E.g. Support Vector Machines (SVMs)
  - Best understood for 'Supervised Learning'
  - Valid for bascially all machine learning algorithms

- Theoretical background used to solve 'A learning problem'
  - Inferring one 'target function' that maps between input and output
  - Learned function can be used to predict output from future input (fitting existing data is not enough)

Unknown Target Function

$$f : X \rightarrow Y$$

(ideal function)

Practical Machine Learning Overview

Unknown Target Function

$$f : X \rightarrow Y$$

(ideal function)

Elements we not exactly (need to) know

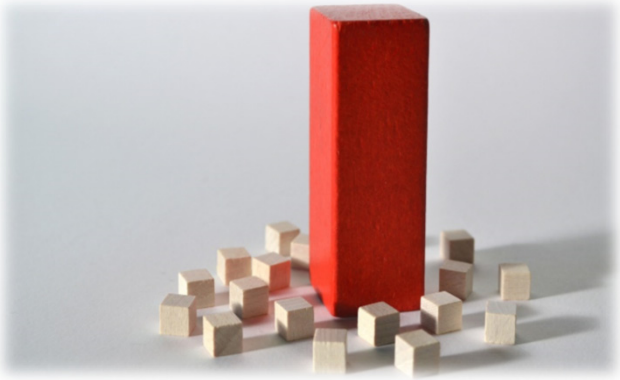
Elements we must and/or should have and that might raise huge demands for storage

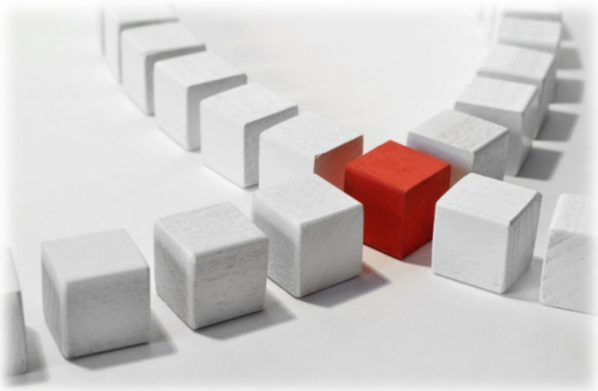
Elements that we derive from our skillset and that can be computationally intensive

Elements that we derive from our skillset

Summary Terminologies & Importance of Theory of Generalization

- Target Function
  - Ideal function that ‘explains‘ the data we want to learn
- Labelled Dataset (samples)
  - ‘in-sample‘ data given to us:
- Learning vs. Memorizing
  - The goal is to create a system that works well ‘out of sample‘
  - In other words we want to classify ‘future data‘ (ouf of sample) correct
- Dataset Part One: Training set
  - Used for training a machine learning algorithms
  - Result after using a training set: a trained system
- Dataset Part Two: Test set
  - Used for testing whether the trained system might work well
  - Result after using a test set: accuracy of the trained model





Learning Approaches – Supervised Learning – Revisited

- Each observation of the predictor measurement(s) has an associated response measurement:
  - Input $\mathbf{x} = x_1, ..., x_d$
  - Output $y_i, i = 1, .., n$
  - Data $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$
  - (the output guides the learning process as a 'supervisor')

- Goal: Fit a model that relates the response to the predictors
  - Prediction: Aims of accurately predicting the response for future observations
  - Inference: Aims to better understanding the relationship between the response and the predictors

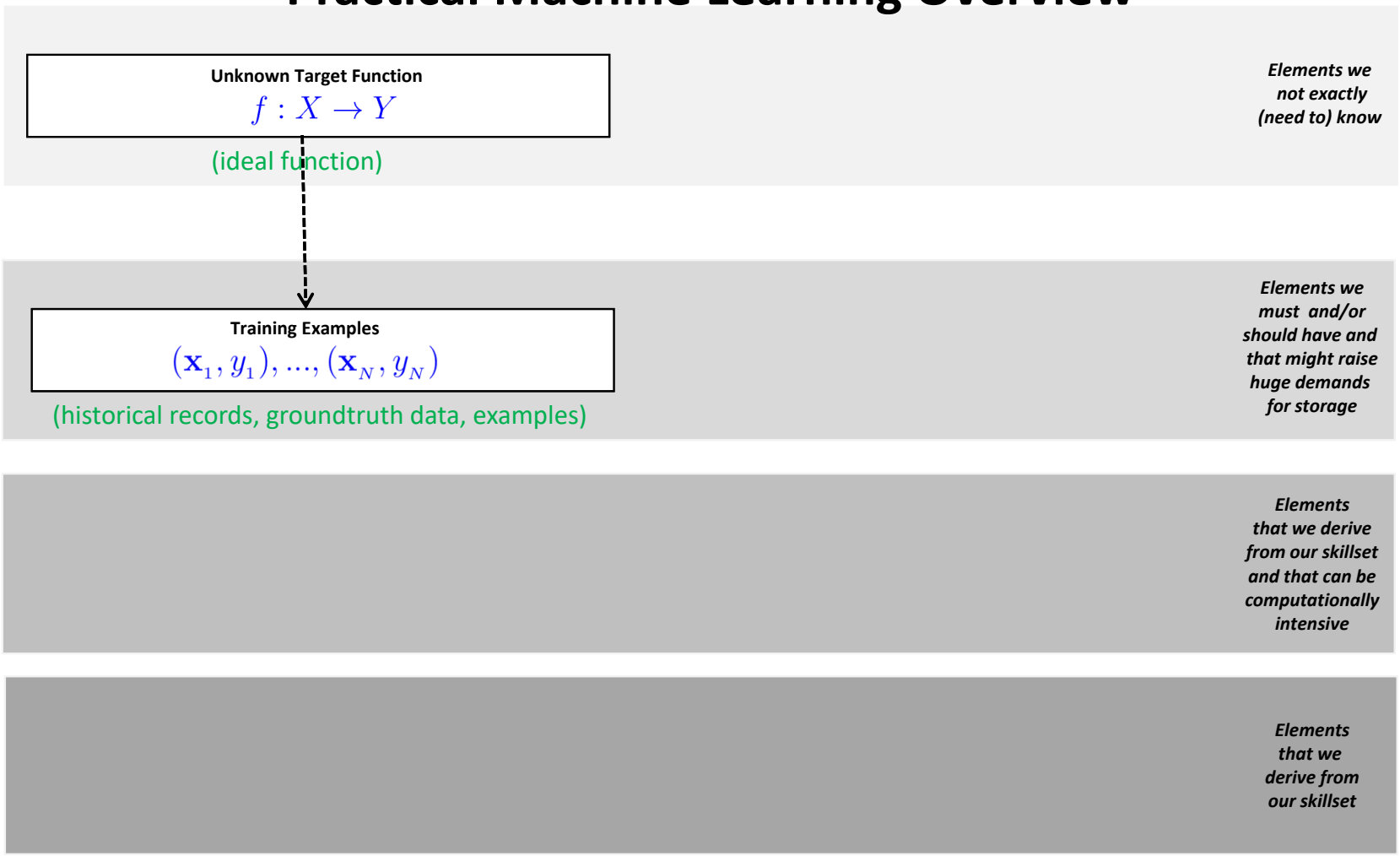
- **Supervised learning approaches fits a model that related the response to the predictors**
- **Supervised learning approaches are used in classification algorithms such as SVMs**
- **Supervised learning works with data = [input, correct output]**

Training Examples

$(\mathbf{x}_1, y_1), ..., (\mathbf{x}_N, y_N)$

(historical records, groundtruth data, examples)

Practical Machine Learning Overview



Step 4: Create Spark Session & Check Application Dataset (1)

```
In [1]: from pyspark.ml import Pipeline
        from pyspark.ml.classification import LogisticRegression
        from pyspark.ml.feature import HashingTF, Tokenizer
        from pyspark.sql import Row
        from pyspark.sql.functions import UserDefinedFunction, col
        from pyspark.sql.types import *
```

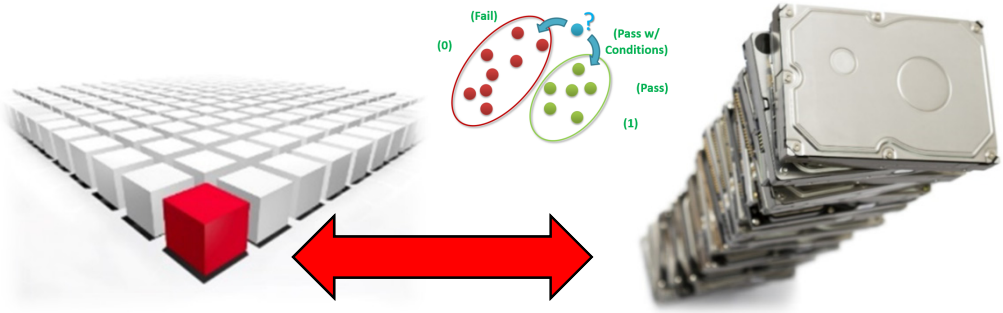
```
Starting Spark application
```

ID	YARN Application ID	Kind	State	Spark UI	Driver log	Current session?
0	application_1536508990334_0003	pyspark	idle	Link	Link	✔

```
SparkSession available as 'spark'.
```

(remember that even idle SparkSessions and Azure HDInsight Clusters cost per minute)

- The execution of the import cell using PySpark kernel notebooks automatically generates in the background a SparkSession with a SparkContext in Azure HDInsight Clusters
- The startup and printout of the Spark application status might take a while when executing the cell (shift+enter)
- The process is the same for any amount of Spark cluster nodes that are used in the deployment process and managed via YARN



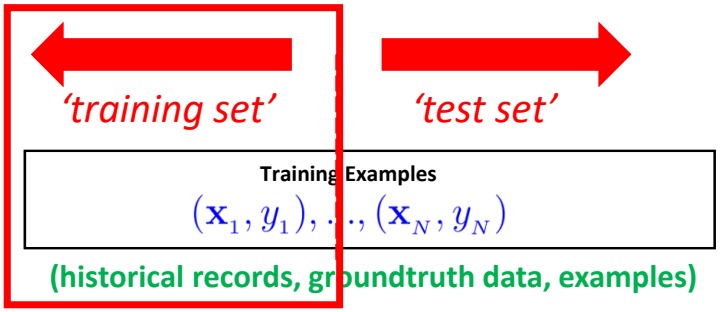
[1] Apache Spark

(the size of the required Spark cluster often depends on the amount of 'big data' or its processing complexity)

- pyspark is a Python Application Programming Interface (API) for Spark available in Microsoft Azure HDInsight Clusters
- Import required libraries for our logistic regression application via selected libraries from pyspark and the Spark Machine Learning Library (MLlib)
- Import selected libraries for better analyzing data & creating features using the Apache Spark Structured Query Language (SQL) library
- Yet Another Resource Negotiator (YARN) is the resource manager user by Apache Hadoop and Apache Spark Big Data Analytics Technologies

Step 4: Create Spark Session & Check Application Dataset (2)

- Load Dataset
  - Read/load dataset from csv file
  - We first will work on the training set
  - There are two files: one training set – one test set




```
In [*]: inspections = spark.read.csv('wasb:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections1.csv', inferSchema=True)
```

(press 'shift + enter' and * appears means it is still processing the request)

```
In [2]: inspections = spark.read.csv('wasb:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections1.csv', inferSchema=True)
```

(inferSchema=True instructs Spark to automatically infer the data type for each column when reading csv files)

- **In Microsoft Azure HDInsight Clusters the SparkContext 'spark' can be used to read/load datasets (e.g. csv: comma-separated values) in order to generate an input Spark dataframe for further use**
- **The dataset location is in the cloud using an Windows Azure Storage Blob (WASB) that is a general-purpose object store – the FoodInspectionData is sample data available by default on the clusters**

Step 4: Create Spark Session & Check Application Dataset (3)

- Perform Feature Selection
  - Not all columns (aka ‘features‘) are needed from schema (i.e. dataframe)
  - Delete not required columns for saving memory when caching the dataset
  - Rename columns in order to get a better organization for the dataset

In [3]:
```
inspections.printSchema()
```

```
root
 |-- _c0: integer (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: integer (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)
 |-- _c9: integer (nullable = true)
 |-- _c10: string (nullable = true)
 |-- _c11: string (nullable = true)
 |-- _c12: string (nullable = true)
 |-- _c13: string (nullable = true)
 |-- _c14: double (nullable = true)
 |-- _c15: double (nullable = true)
 |-- _c16: string (nullable = true)
```

In [4]:
```
# Helper function to drop unused columns and rename interesting columns.
def selectInterestingColumns(rawDf):
    # Mapping column index to name.
    columnNames = {0: "id", 1: "name", 12: "results", 13: "violations"}

    # Rename column from '_c{id}' to something meaningful.
    cols = [col(rawDf.columns[i]).alias(columnNames[i]) for i in columnNames.keys()]

    # Drop columns we are not using.
    df = rawDf.select(cols)

    # Replace null in column 'violations' with empty string.
    # We are going to run LogisticRegression on this column and it doesn't like null.
    return df.fillna('', ["violations"])
```

(creates another dataframe)

In [5]:
```
df = selectInterestingColumns(inspections).cache()

# Do a count to cache the whole dataframe in memory
df.count()
```

```
13101
```

Step 4: Create Spark Session & Check Application Dataset (4)

- Perform Data Exploration
  - Get a better understanding of the dataset
  - Features are ID, Name, Results, Violations
  - Printout of one row helps to understand the data better

```
In [6]: df.printSchema()
```

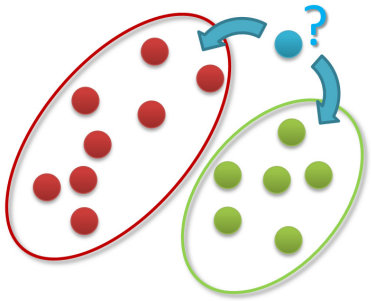
```
root
 |-- id: integer (nullable = true)
 |-- name: string (nullable = true)
 |-- results: string (nullable = true)
 |-- violations: string (nullable = false)
```

(schema of newly created dataframe)

```
In [7]: df.take(1)
```

```
[Row(id=413707, name=u'LUNA PARK INC', results=u'Fail', violations=u'24. DISH WASHING FACILITIES: PROPERLY DESIGNED, CONSTRUCTED, MAINTAINED, INSTALLED, LOCATED AND OPERATED - Comments: All dishwashing machines must be of a type that complies with all requirements of the plumbing section of the Municipal Code of Chicago and Rules and Regulation of the Board of Health. OBSEVERD THE 3 COMPARTMENT SINK BACKING UP INTO THE 1ST AND 2ND COMPARTMENT WITH CLEAR WATER AND SLOWLY DRAINING OUT. INST NEED HAVE IT REPAIR. CITATION ISSUED, SERIOUS VIOLATION 7-38-030 H000062369-10 COURT DATE 10-28-10 TIME 1 P.M. ROOM 107 400 W. SURPERIOR. | 36. LIGHTING: REQUIRED MINIMUM FOOT-CANDLES OF LIGHT PROVIDED, FIXTURES SHIELDED - Comments: Shielding to protect against broken glass falling into food shall be provided for all artificial lighting sources in preparation, service, and display facilities. LIGHT SHIELD ARE MISSING UNDER HOOD OF  COOKING EQUIPMENT AND NEED TO REPLACE LIGHT UNDER UNIT. 4 LIGHTS ARE OUT IN THE REAR CHILDREN AREA,IN THE KINDERGARDEN CLASS ROOM. 2 LIGHT ARE OUT EAST REAR, LIGHT FRONT WEST ROOM. NEED TO REPLACE ALL LIGHT THAT ARE NOT WORKING. | 35. WALLS, CEILINGS, ATTACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAIR, SURFACES CLEAN AND DUST-LESS CLEANING METHODS - Comments: The walls and ceilings shall be in good repair and easily cleaned. MISSING CEILING TILES WITH STAINS IN WEST,EAST, IN FRONT AREA WEST, AND BY THE 15MOS AREA. NEED TO BE REPLACED. | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND MAINTAINED - Comments: All food and non-food contact equipment and utensils shall be smooth, easily cleanable, and durable, and shall be in good repair. SPLASH GUARDED ARE NEEDED BY THE EXPOSED HAND SINK IN THE KITCHEN AREA | 34. FLOORS: CONSTRUCTED PER CODE, CLEANED, GOOD REPAIR, COVING INSTALLED, DUST-LESS CLEANING METHODS USED - Comments: The floors shall be constructed per code, be smooth and easily cleaned, and be kept clean and in good repair. INST NEED TO ELEVATE ALL FOOD ITEMS 6INCH OFF THE FLOOR 6 INCH AWAY FORM WALL.  ')]
```

Classification Problem



Step 4: Create Spark Session & Check Application Dataset (5)

- Data Preparation & Optimization
  - Use registerTempTable() to create an in-memory table in the cluster
  - In-memory is fast for data analysis (cf. Lecture 1) – optimization!
  - Use Hive's highly-optimized column format
  - Data is ready for our analysis steps – five samples shown below

[1] Apache Spark

```
In [8]: df.registerTempTable('CountResults')
```

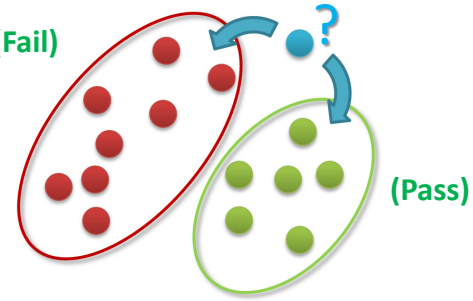
```
In [9]: df.show(5)
```

```
+------+--------------------+-------+--------------------+
|    id|                name|results|          violations|
+------+--------------------+-------+--------------------+
|413707|       LUNA PARK INC|   Fail|24. DISH WASHING ...|
|391234|       CAFE SELMARIE|   Fail|2. FACILITIES TO ...|
|413751|          MANCHU WOK|   Pass|33. FOOD AND NON-...|
|413708|BENCHMARK HOSPITA...|   Pass|                    |
|413722|           JJ BURGER|   Pass|                    |
+------+--------------------+-------+--------------------+
only showing top 5 rows
```

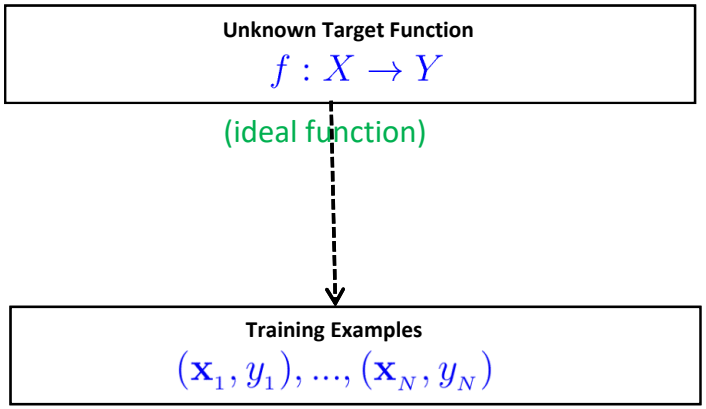
(Really two Classes?)

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

Binary Classification Problem



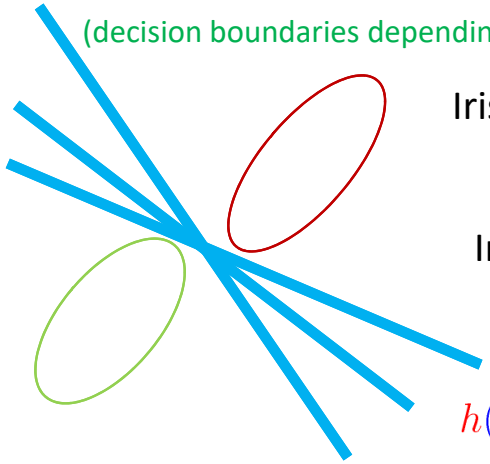
Mathematical Building Blocks – Our Linear Perceptron Example



(historical records, groundtruth data, examples)

1. Some pattern exists
2. No exact mathematical formula (i.e. target function)
3. Data exists

(if we would know the exact target function we dont need machine learning, it would not make sense)



(decision boundaries depending on f)

Iris-virginica if $\sum_{i=1}^{d} w_i x_i > threshold$

Iris-setosa if $\sum_{i=1}^{d} w_i x_i < threshold$

(w_i and threshold are still unknown to us)

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$

(we search a function similiar like a target function)

Different Models – Hypothesis Set to Choose From → Logistic Regression

Hypothesis Set

$$\mathcal{H} = \{h\};\ g \in \mathcal{H}$$

$$\mathcal{H} = \{h_1, \ldots, h_m\};$$

(all candidate functions
derived from models
and their parameters)

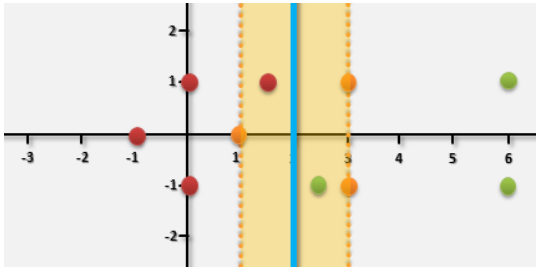
- **Choosing from various model approaches h_1, …, h_m is a different hypothesis**
- **Additionally a change in model parameters of h_1, …, h_m means a different hypothesis too**

'select one function'
that best approximates

Final Hypothesis

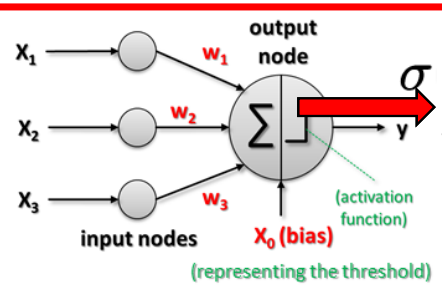
$$g \approx f$$

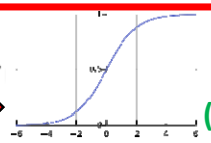
h_1



(e.g. support vector machine model)

h_2





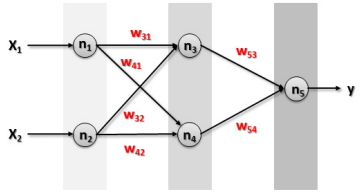
$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

(model function)

$$h(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

(e.g. logistic regression model)

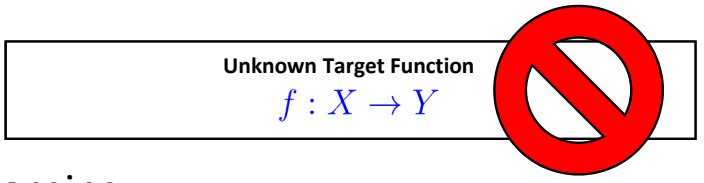
h_m



(e.g. artificial neural network model)

Feasibility of Learning – Hypothesis Set & Final Hypothesis

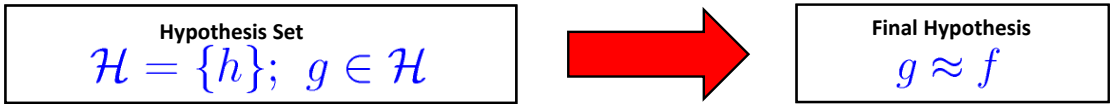
- The ‘ideal function‘ will remain unknown in learning
  - Impossible to know and learn from data
  - If known a straightforward implementation would be better than learning
  - E.g. hidden features/attributes of data not known or not part of data



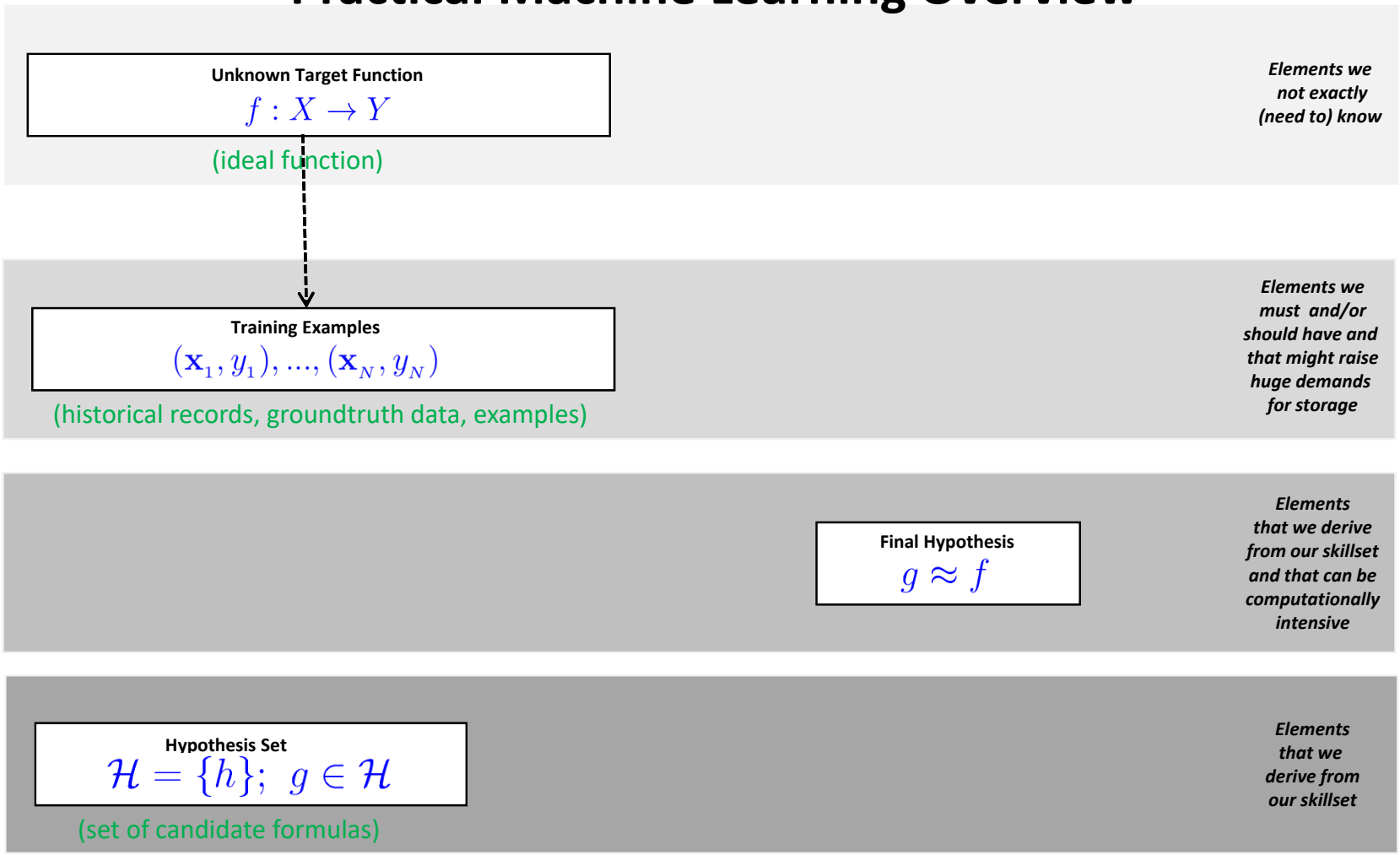
- But ‘(function) approximation‘ of the target function is possible
  - Use training examples to learn and approximate it
  - Hypothesis set $\mathcal{H}$ consists of m different hypothesis (candidate functions)

$$\mathcal{H} = \{h_1, ..., h_m\};$$

‘select one function‘ that best approximates $g : X \rightarrow Y$

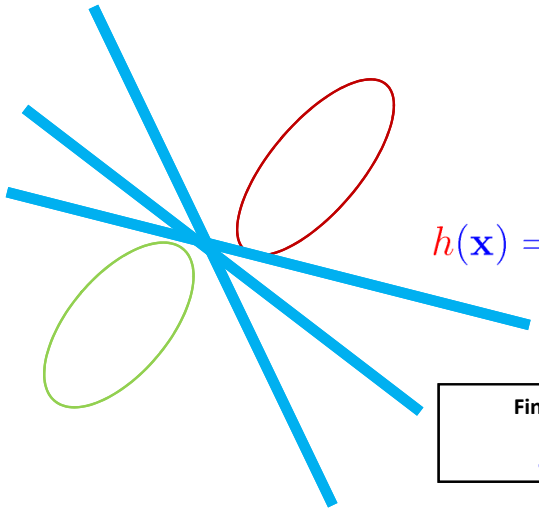


Practical Machine Learning Overview



Mathematical Building Blocks – Our Linear Perceptron Example

(decision boundaries depending on f)



$$\mathcal{H} = \{h_1, ..., h_m\};$$

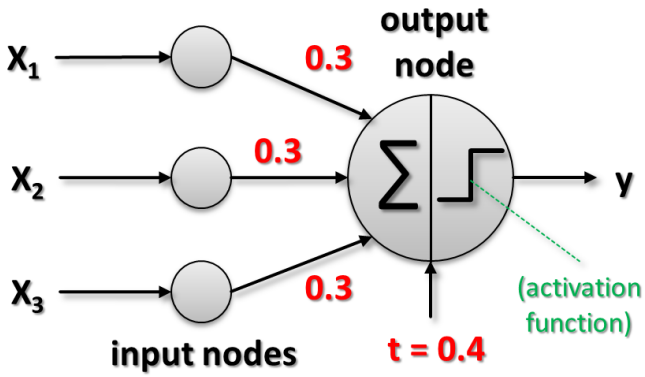
(we search a function similiar like a target function)

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$

Final Hypothesis

$$g \approx f$$

$$h(\mathbf{x}) = sign\left(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold\right); h \in \mathcal{H}$$



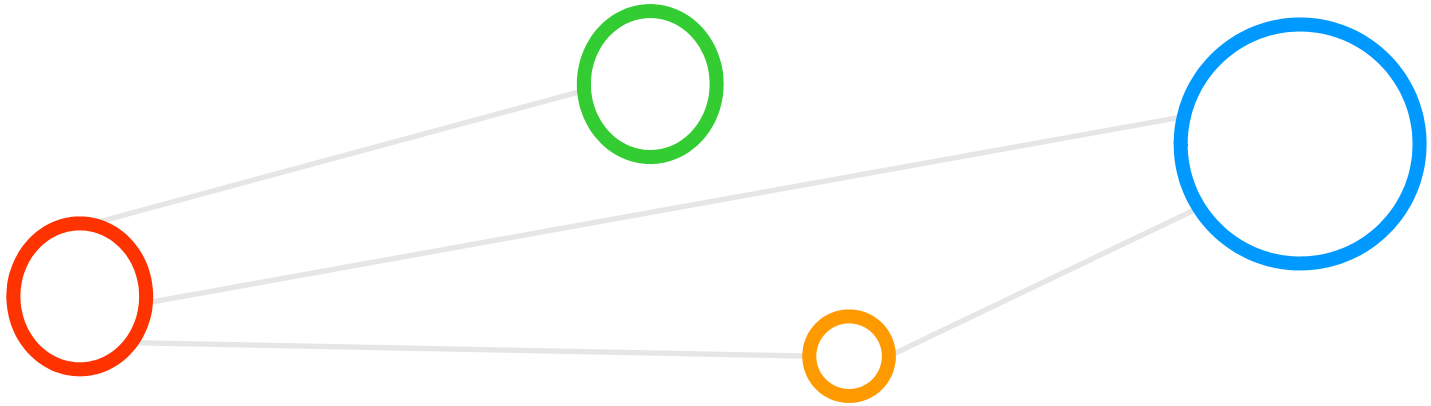
Hypothesis Set

$$\mathcal{H} = \{h\};\ g \in \mathcal{H}$$

(Perceptron model – linear model)

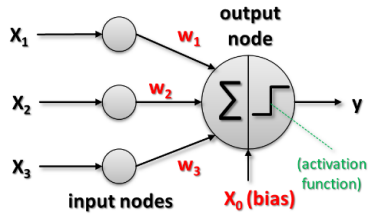
(trained perceptron model and our selected final hypothesis)

Machine learning via Cloud Demo using Apache Spark Pipelines



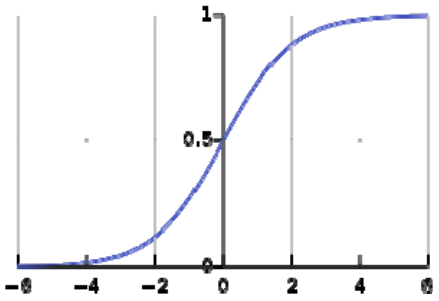
More Advanced: Logistic Regression Using Non-Linear Activation Function

- Linear Classification
  - Simple binary classification (linearly seperable)
  - Linear combination of the inputs x_i with weights w_i
- Linear Regression
  - Real value with the activiation being the identity function
  - E.g. how much sales given marketing money spend on TV advertising
- Logistic Regression
  - Model/error measure/learning algorithm is different
  - Captures non-linear data dependencies using the so-called Sigmoid function
  - Key idea is to bring values between 0 and 1 to estimate a probability
  - (candidate model for pass/fail in our application)



$$\mathcal{H} = \{h_1, \ldots, h_m\};$$

(we search a function similiar like a target function)



(model function)

$$h(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

[1] Apache Spark

Step 5: Initial Data Analysis, Visualization & Modeling (1)

- Initial data analysis
  - How are results encoded in values and what do they mean for our model?
  - What is the ratio of the results across all data samples?

```
In [10]: df.select('results').distinct().show()
```

```
+--------------------+
|             results|
+--------------------+
|                Fail|
|Business Not Located|
|  Pass w/ Conditions|
|     Out of Business|
|                Pass|
+--------------------+
```

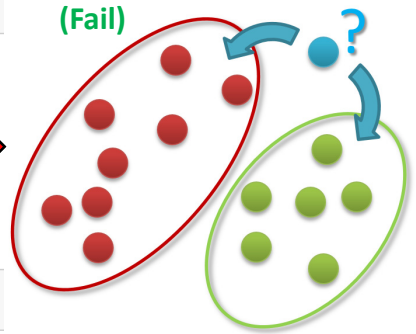
(Five Classes?!)

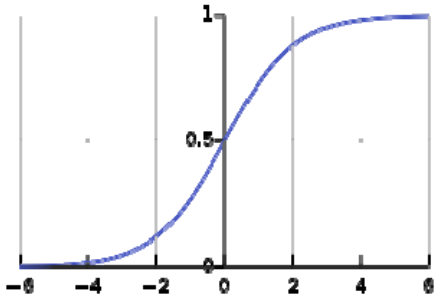
```
In [11]: %%sql -o count_results_df
SELECT results, COUNT(results) AS cnt FROM CountResults GROUP BY results ORDER BY cnt DESC
```

Type: Table | Pie | Scatter | Line | Area | Bar

results	cnt
Pass	8457
Fail	3324
Pass w/ Conditions	1033
Out of Business	281
Business Not Located	6

Binary Classification Problem





(model function)

$h(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$

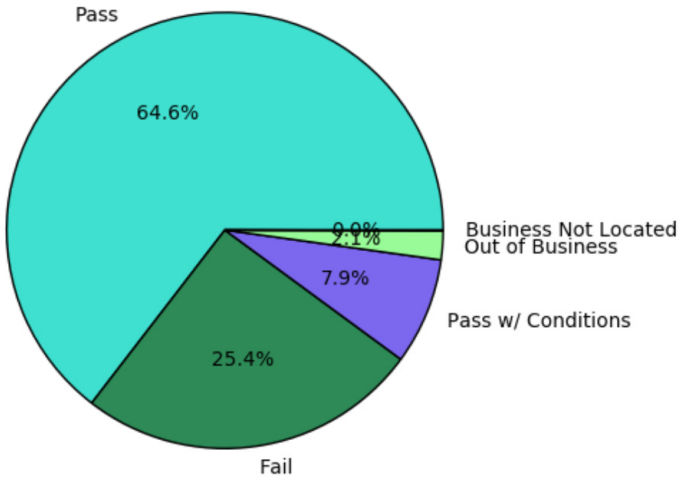
- **An initial data analysis or data inspection often reveals in machine learning practice that the data must be especially prepared or modified to work with a specific machine learning model**
- **The process of preparing data for machine learning models is called data preprocessing or data preparation and often includes aspects like data cleaning, feature selection, sub-sampling, etc.**
- **It is important to understand the relevant data in question to understand what the machine learning model is really doing while 'learning from data': 'rubbish-in' means 'rubbish-out'**

Step 5: Initial Data Analysis, Visualization & Modeling (2)

In [12]:
```
%%local
%matplotlib inline
import matplotlib.pyplot as plt


labels = count_results_df['results']
sizes = count_results_df['cnt']
colors = ['turquoise', 'seagreen', 'mediumslateblue', 'palegreen', 'coral']
plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors)
plt.axis('equal')
```

(The plot is created from locally persisted count_results_df dataframe, therefore code snippet must begin with the %%local 'magic' & will be executed locally on Jupyter server)

Out[12]:
```
(-1.0119403344538858,
 1.0000000000185918,
 -1.008810095783151,
 1.0072197845427724)
```



- Data visualization as part of the data exploration or initial data analysis is a common way to get a better understanding of the dataset and to perform several optimizations, tunings, or refinements
- Matplotlib is a library available in Python that can be used to create meaningful data visualizations
- Infographics are a compact visualization of facts often used in conjunction with data analysis results or machine learning

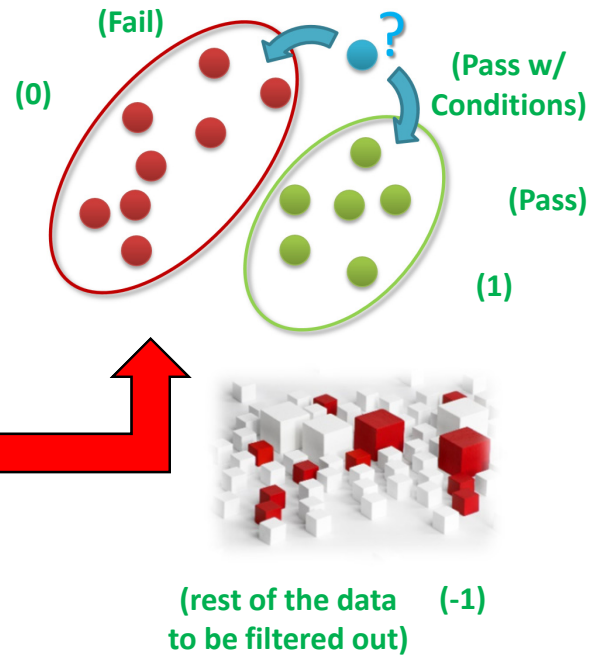
Step 5: Initial Data Analysis, Visualization & Modeling (3)

- Modeling
  - Apply model of logistic regression with feature (violations) and class label
  - Visualization shows ‘results‘ has five different outcomes (aka five classes)
  - Binary classification model needs binary labels (aka ‘results‘, two classes)
  - Setup two classes: Fail and Pass (including those ‘Pass w/Conditions‘)
  - Remove other results (‘Business Not Located‘ and ‘Out of Business‘)
  - Implications to learning from dataset: ok since removed ‘classes‘ together are a tiny fraction of the dataset (cf. Visualization)
  - Model labels: 0.0 (fail) and 1.0 (pass) and -1.0 (other, later filtered out)

```
In [13]: # Note: We can do the same thing with pyspark.sql.functions.when(). It doesn't use UDF and is faster.
         # However, we would like to demonstrate UserDefinedFunction here as an example.
         def labelForResults(s):
             if s == 'Fail':
                 return 0.0
             elif s == 'Pass w/ Conditions' or s == 'Pass':
                 return 1.0
             else:
                 return -1.0
         label = UserDefinedFunction(labelForResults, DoubleType())
         labeledData = df.select(label(df.results).alias('label'), df.violations).where('label >= 0')
```

- **The process of data preparation with filtering data out is also often called sampling or sub-sampling and can be dangerous: e.g., creating an unbalanced class problem to learn from**

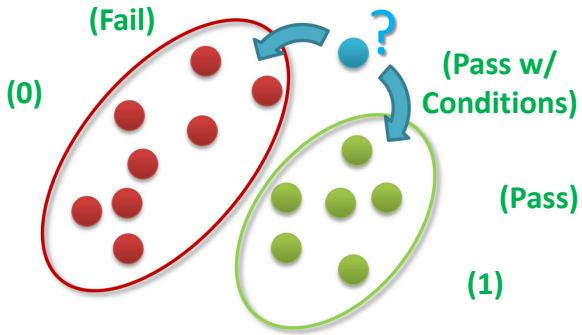
Binary Classification Problem



Step 5: Initial Data Analysis, Visualization & Modeling (4)

- Modeling
  - Check labelled data setup and start the modeling process
  - Feature: violations
    (full text, but machines learn on numbers → next step)
  - Class label (0.0, 1.0)
  - (modeling decision is that we disregard some of the samples,
    this might influence the quality of the modeling process → here ok)

Binary Classification Problem needs a Machine Learning Model that fits

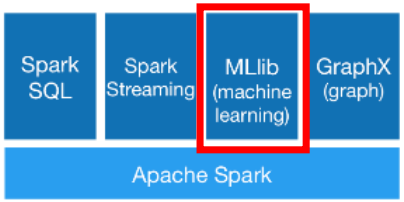



```
In [14]: labeledData.take(1)
```

```
[Row(label=0.0, violations=u'24. DISH WASHING FACILITIES: PROPERLY DESIGNED, CONSTRUCTED, MAINTAINED, INSTALLED, LOCATED AND OP
ERATED - Comments: All dishwashing machines must be of a type that complies with all requirements of the plumbing section of th
e Municipal Code of Chicago and Rules and Regulation of the Board of Health. OBSEVERD THE 3 COMPARTMENT SINK BACKING UP INTO TH
E 1ST AND 2ND COMPARTMENT WITH CLEAR WATER AND SLOWLY DRAINING OUT. INST NEED HAVE IT REPAIR. CITATION ISSUED, SERIOUS VIOLATIO
N 7-38-030 H000062369-10 COURT DATE 10-28-10 TIME 1 P.M. ROOM 107 400 W. SURPERIOR. | 36. LIGHTING: REQUIRED MINIMUM FOOT-CANDL
ES OF LIGHT PROVIDED, FIXTURES SHIELDED - Comments: Shielding to protect against broken glass falling into food shall be provid
ed for all artificial lighting sources in preparation, service, and display facilities. LIGHT SHIELD ARE MISSING UNDER HOOD OF
COOKING EQUIPMENT AND NEED TO REPLACE LIGHT UNDER UNIT. 4 LIGHTS ARE OUT IN THE REAR CHILDREN AREA,IN THE KINDERGARDEN CLASS RO
OM. 2 LIGHT ARE OUT EAST REAR, LIGHT FRONT WEST ROOM. NEED TO REPLACE ALL LIGHT THAT ARE NOT WORKING. | 35. WALLS, CEILINGS, AT
TACHED EQUIPMENT CONSTRUCTED PER CODE: GOOD REPAIR, SURFACES CLEAN AND DUST-LESS CLEANING METHODS - Comments: The walls and cei
lings shall be in good repair and easily cleaned. MISSING CEILING TILES WITH STAINS IN WEST,EAST, IN FRONT AREA WEST, AND BY TH
E 15MOS AREA. NEED TO BE REPLACED. | 32. FOOD AND NON-FOOD CONTACT SURFACES PROPERLY DESIGNED, CONSTRUCTED AND MAINTAINED - Com
ments: All food and non-food contact equipment and utensils shall be smooth, easily cleanable, and durable, and shall be in goo
d repair. SPLASH GUARDED ARE NEEDED BY THE EXPOSED HAND SINK IN THE KITCHEN AREA | 34. FLOORS: CONSTRUCTED PER CODE, CLEANED, G
OOD REPAIR, COVING INSTALLED, DUST-LESS CLEANING METHODS USED - Comments: The floors shall be constructed per code, be smooth a
nd easily cleaned, and be kept clean and in good repair. INST NEED TO ELEVATE ALL FOOD ITEMS 6INCH OFF THE FLOOR 6 INCH AWAY FO
RM WALL.  ')]
```

Apache Spark Machine Learning Library (MLlib) – Select Algorithm for Problem

- Classification algorithms
  - include logistic regression and naïve Bayes
- Regression
  - Generalized linear regression and survival regression
- Tree-based approaches
  - Standard decision trees, random forests, gradient-boosted trees
- Alternating least squares (ALS)
  - Used to create recommendation engines



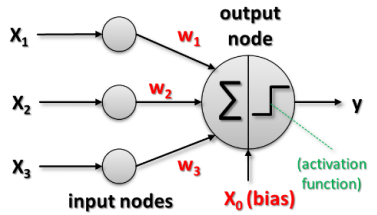
Spark MLlib

[1] Apache Spark

- Clustering
  - K-Means and Gaussian Mixture Models (GMM)
- Pattern mining
  - Frequent itemsets & Association rule mining

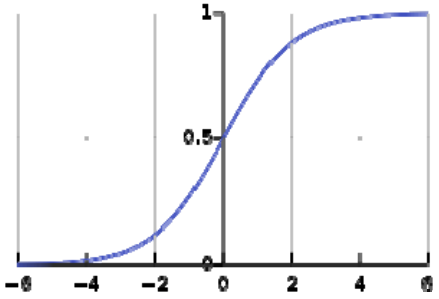
Logistic Regression Using Non-Linear Activation Function – Revisited

- Linear Classification
  - Simple binary classification (linearly seperable)
  - Linear combination of the inputs x_i with weights w_i



- Linear Regression
  - Real value with the activiation being the identity function
  - E.g. how much sales given marketing money spend on TV advertising

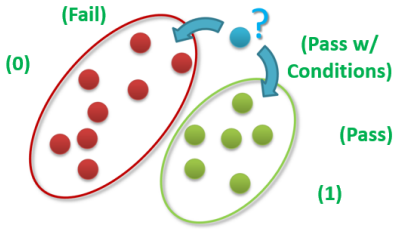
- Logistic Regression
  - Model/error measure/learning algorithm is different
  - Captures non-linear data dependencies using the so-called Sigmoid function
  - Key idea is to bring values between 0 and 1 to estimate a probability
  - (candidate model for pass/fail in our application)



(model function)

$$h(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$



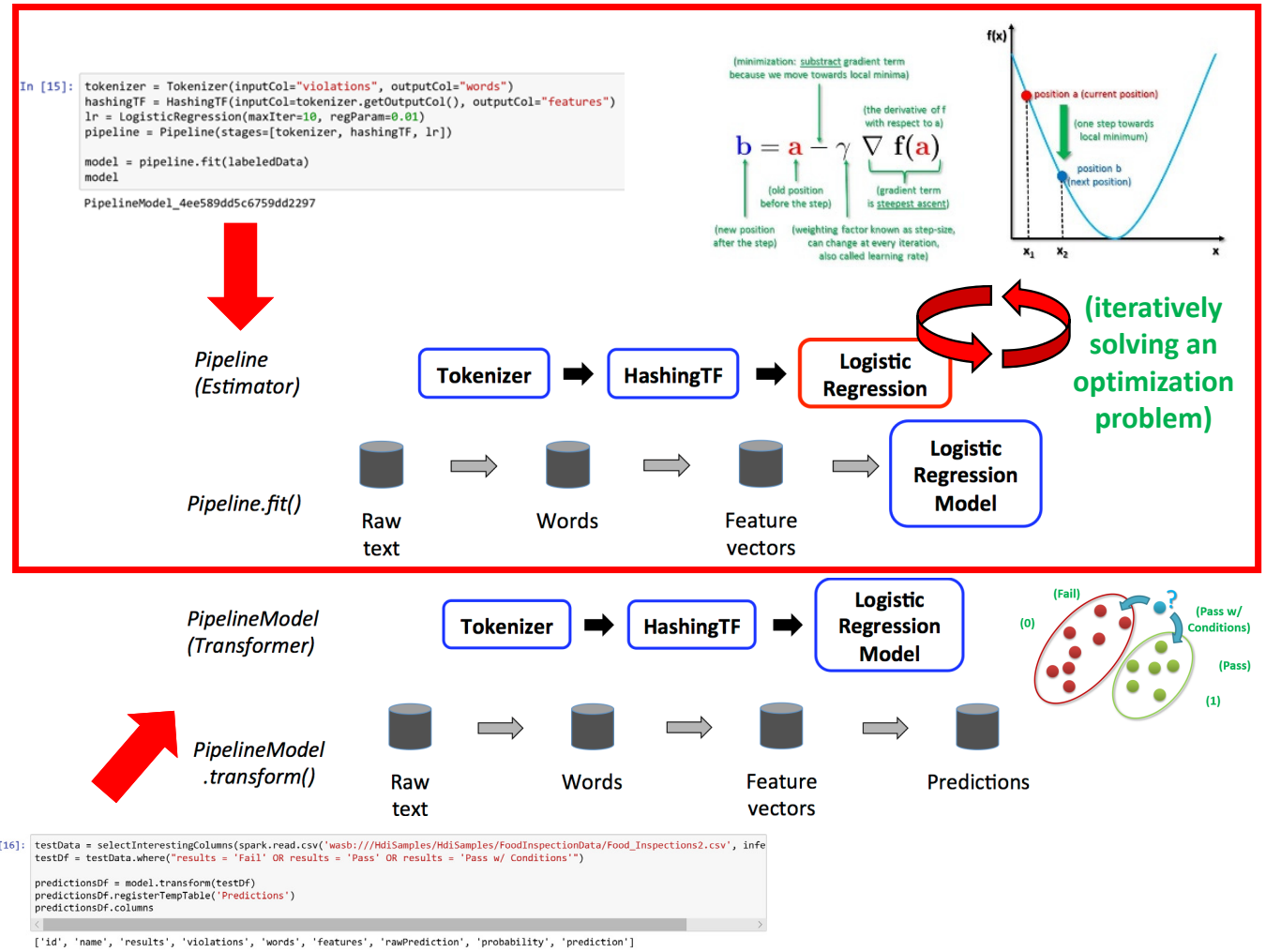

```
lr = LogisticRegression(maxIter=10, regParam=0.01)
```

Apache Spark Pipelines & Logistic Regression Example – Training Process

- Apache Spark Pipeline
  - Includes 'bigger picture' than just one machine learning algorithm

- **A data science pipeline in Apache Spark is a sequence of stages with either an 'Transformer' or an 'Estimator'**
- **Apache Spark pipelines and their stages run in order and the input data (i.e., DataFrame) is transformed as it passes through each stage**
- **A pipeline might consists of stages that perform data preparation steps before the actual machine learning steps begin; in our example the Tokenizer and HashingTF before performing Logistic Regression**
- **A pipeline in the Estimater stage uses the method fit() to iteratively train a PipelineModel (i.e., or fitted Pipeline); in our example a Logistic Regression Model using labeledData from the Chicago Food Inspection application example**
- **The resulting PipelineModel exists within memory and can be used: e.g., testing (later)**

modified from [13] Apache Spark Pipelines



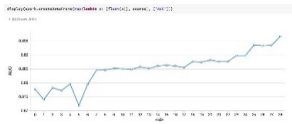
Apache Spark Machine Learning Library (MLlib) & Logistic Regression Example

- Classification as optimization problem
  - E.g., food inspection in restaurants (cf. Lecture 1 & Assignment #1)
  - Using Logistic Regression in Spark
  - Solves inherent optimization problem via optimizer/solver
  - E.g., Stochastic Gradient Descent (SGD) instead of Perceptron Learning Algorithm

(logistic function)

```
1 / (1 + exp(-w^T x))
```

(iteratively solving an optimization problem using training data and testing data)



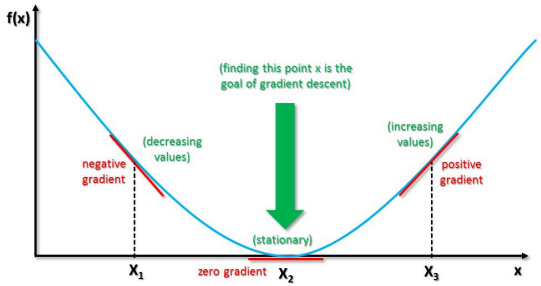
```
log(1 + exp(-yw^T x))
```

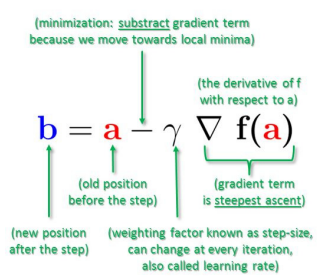
(loss function, logistic loss)

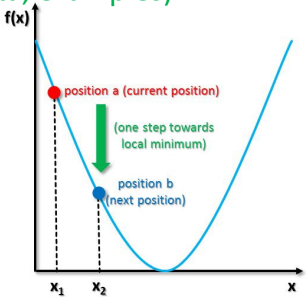
Training Examples

$$(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$$

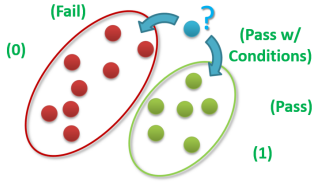
(historical records, groundtruth data, examples)





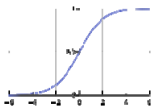


[6] Understanding Parallelization of Machine Learning Algorithms in Apache Spark



$$h(\mathbf{x}) = sign\left(\left(\sum_{i=0}^{d} w_i x_i\right)\right); x_0 = 1$$

$$h(\mathbf{x}) = \sigma\ (\mathbf{w} \cdot \mathbf{x})$$

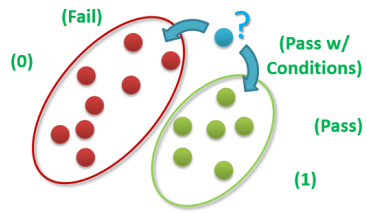


$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

- **Training machine learning is an optimization problem used with many optimizers like stochastic gradient descent (SGD)**
- **The optimization has the goal to learn the coefficients (i.e. weights) based on the examples and fit a line based on the points - Getting towards the optimal model is called convergence**
- **The goal of the optimization is to either maximize a positive metric like the area under the curve (AUC) or minimize a negative metric (e.g., error on testset using a specific error function)**

Step 6: Perform Logistic Regression on Spark Cluster

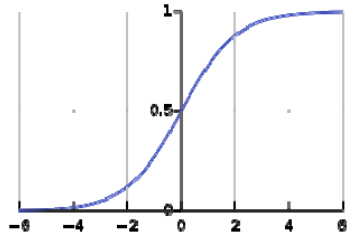
- Supervised learning via logistic regression (label-feature vector)
  - Convert ‘violations‘ semi-structured column to array of real numbers



- **Convert labeled data set into a format that can be trained with a logistic regression model**
- **One approach for processing natural language is to assign each distinct word an ‘index‘ in free-text and then pass a vector to the machine learning algorithm such that each index's value contains the relative frequency of that word in the text string – here represented in the ‘violations‘ text column**
- **Use approach of tokenizer with each ‘violations‘ string for individual words – then HashingTF converts each set of tokens into a feature vector passed to a pipeline fueling a Logistic Regression**

```
In [15]: tokenizer = Tokenizer(inputCol="violations", outputCol="words")
         hashingTF = HashingTF(inputCol=tokenizer.getOutputCol(), outputCol="features")
         lr = LogisticRegression(maxIter=10, regParam=0.01)
         pipeline = Pipeline(stages=[tokenizer, hashingTF, lr])

         model = pipeline.fit(labeledData)
         model

         PipelineModel_4ee589dd5c6759dd2297
```

(The learning steps and label data conversion sequence are part of a pipeline)

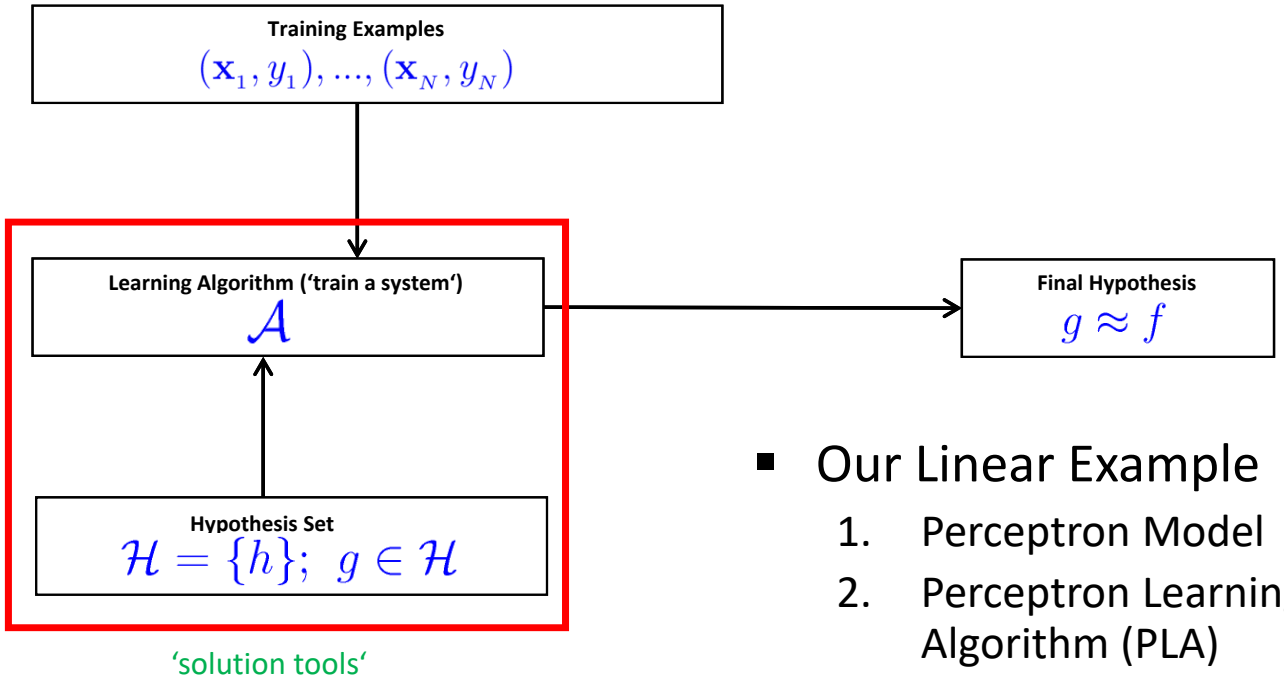


- **The pipeline.fit() command trained on a labeled dataset in order to apply supervised learning using a logistic regression binary classification model that can be used to classify new unseen data now**

➢ **Lecture 3 provides details about Apache Spark and other available Spark MLlib models & pipelines used for machine learning in clouds**

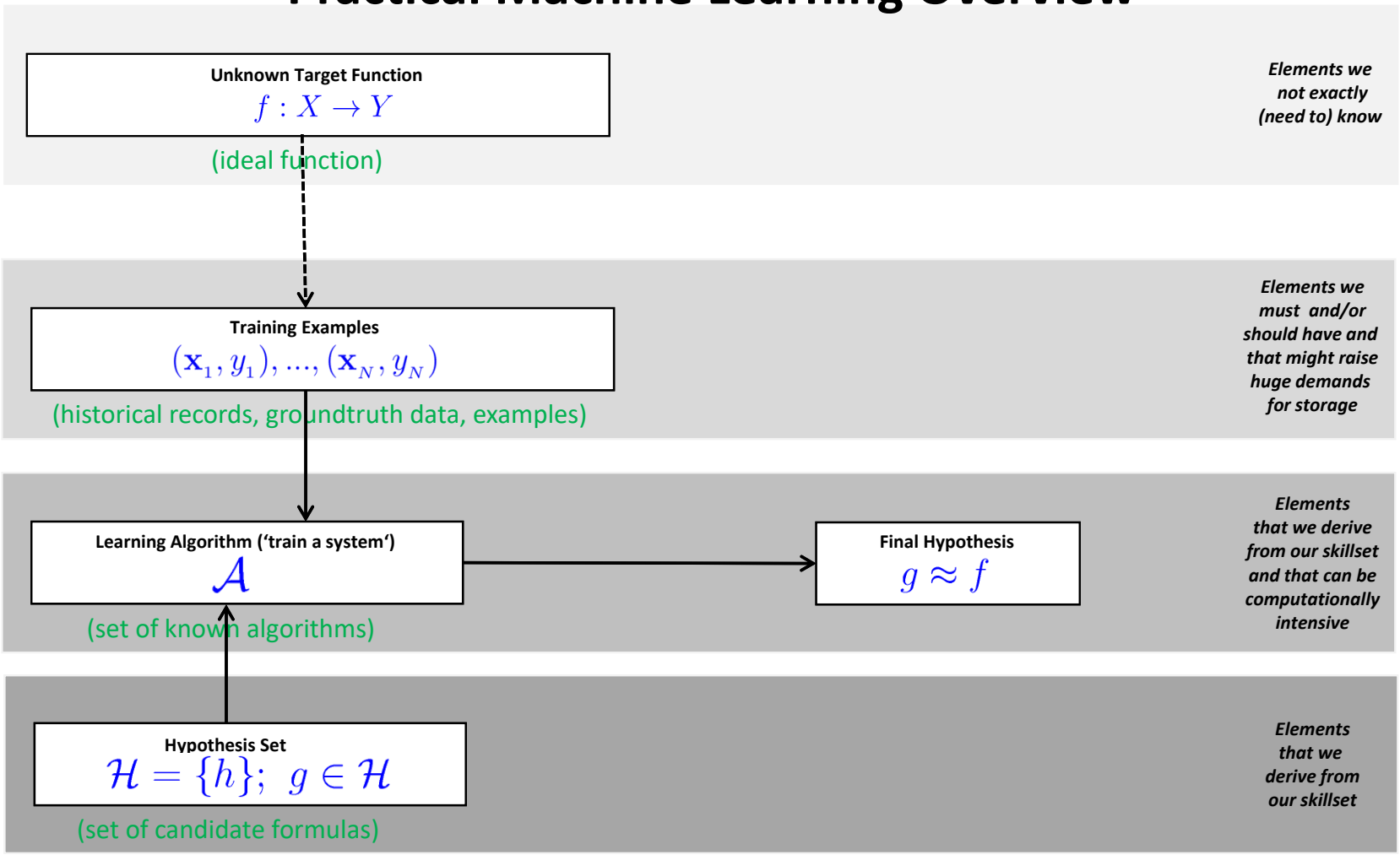
The Learning Model: Hypothesis Set & Learning Algorithm

- The solution tools – the learning model:
  1. Hypothesis set $\mathcal{H}$ - a set of candidate formulas /models
  2. Learning Algorithm $\mathcal{A}$ - 'train a system' with known algorithms

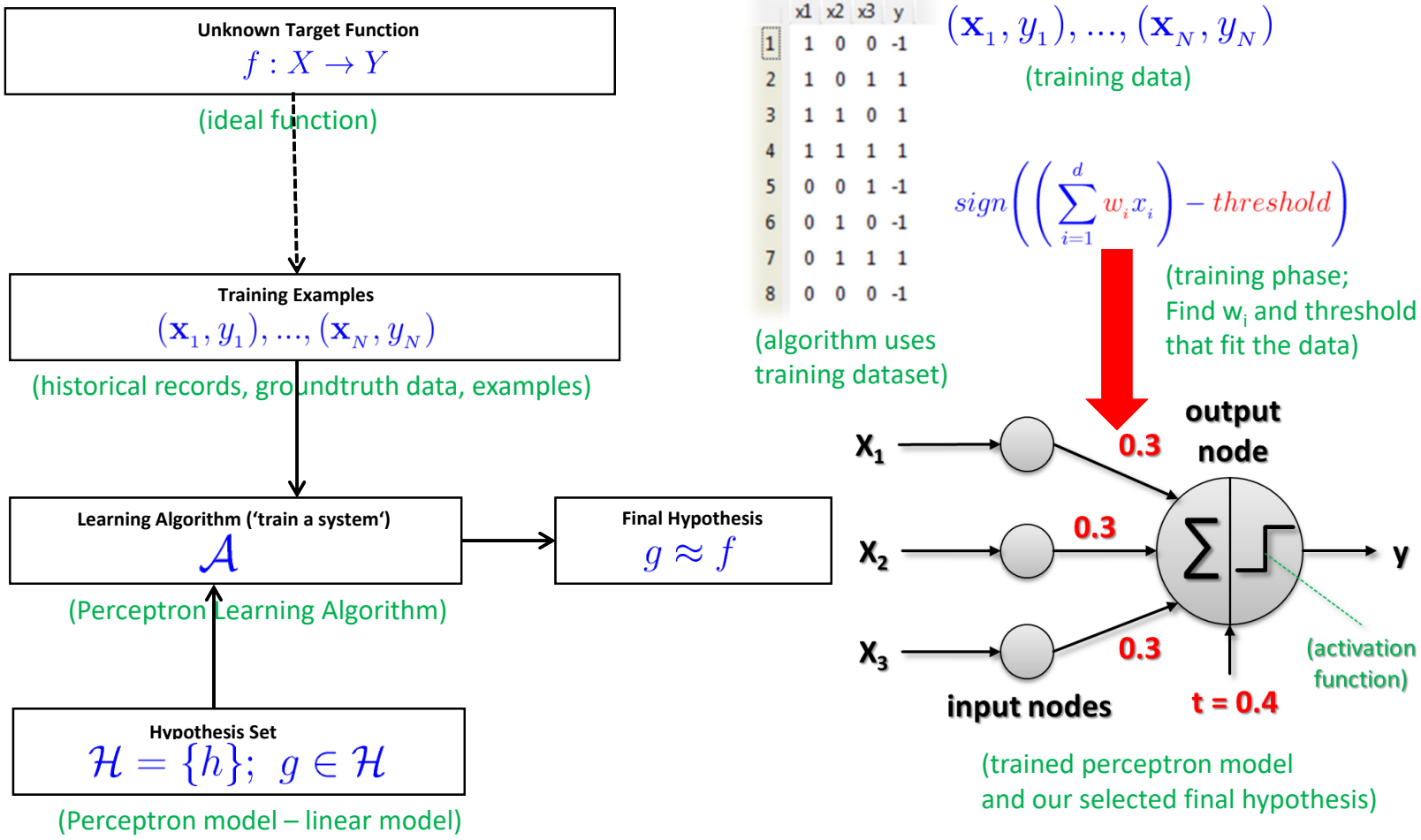


- Our Linear Example
  1. Perceptron Model
  2. Perceptron Learning Algorithm (PLA)

Practical Machine Learning Overview



Mathematical Building Blocks (3) – Our Linear Example



Different Models – Picked Logistic Regression & Loss Function as Error Measure

Hypothesis Set

$$\mathcal{H} = \{h\};\ g \in \mathcal{H}$$

$$\mathcal{H} = \{h_1, \ldots, h_m\};$$

(all candidate functions
derived from models
and their parameters)

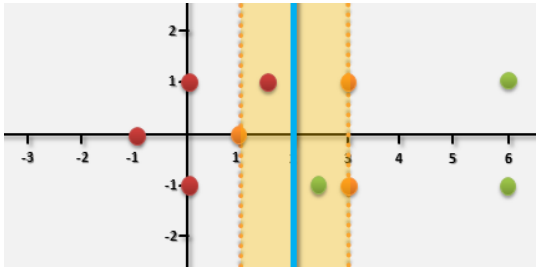
- **Choosing from various model approaches h_1, …, h_m is a different hypothesis**
- **Additionally a change in model parameters of h_1, …, h_m means a different hypothesis too**

'select one function'
that best approximates

Final Hypothesis

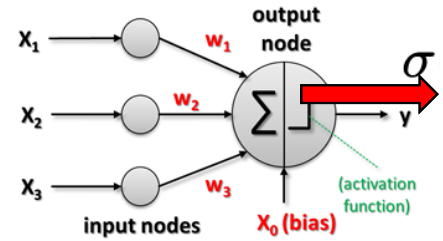
$$g \approx f$$

h_1



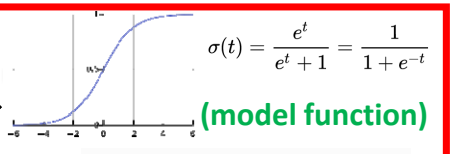
(e.g. support vector machine model)

h_2



(representing the threshold)

(e.g. logistic regression model)



$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

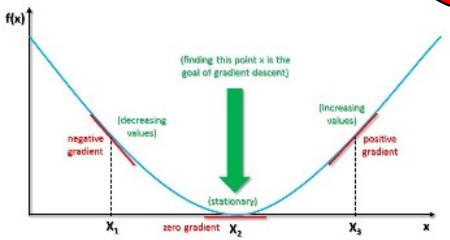
(model function)

```
1 / (1 + exp(-w^T x))
```

$$h(\mathbf{x}) = \sigma(\mathbf{w} \cdot \mathbf{x})$$

Error Measure

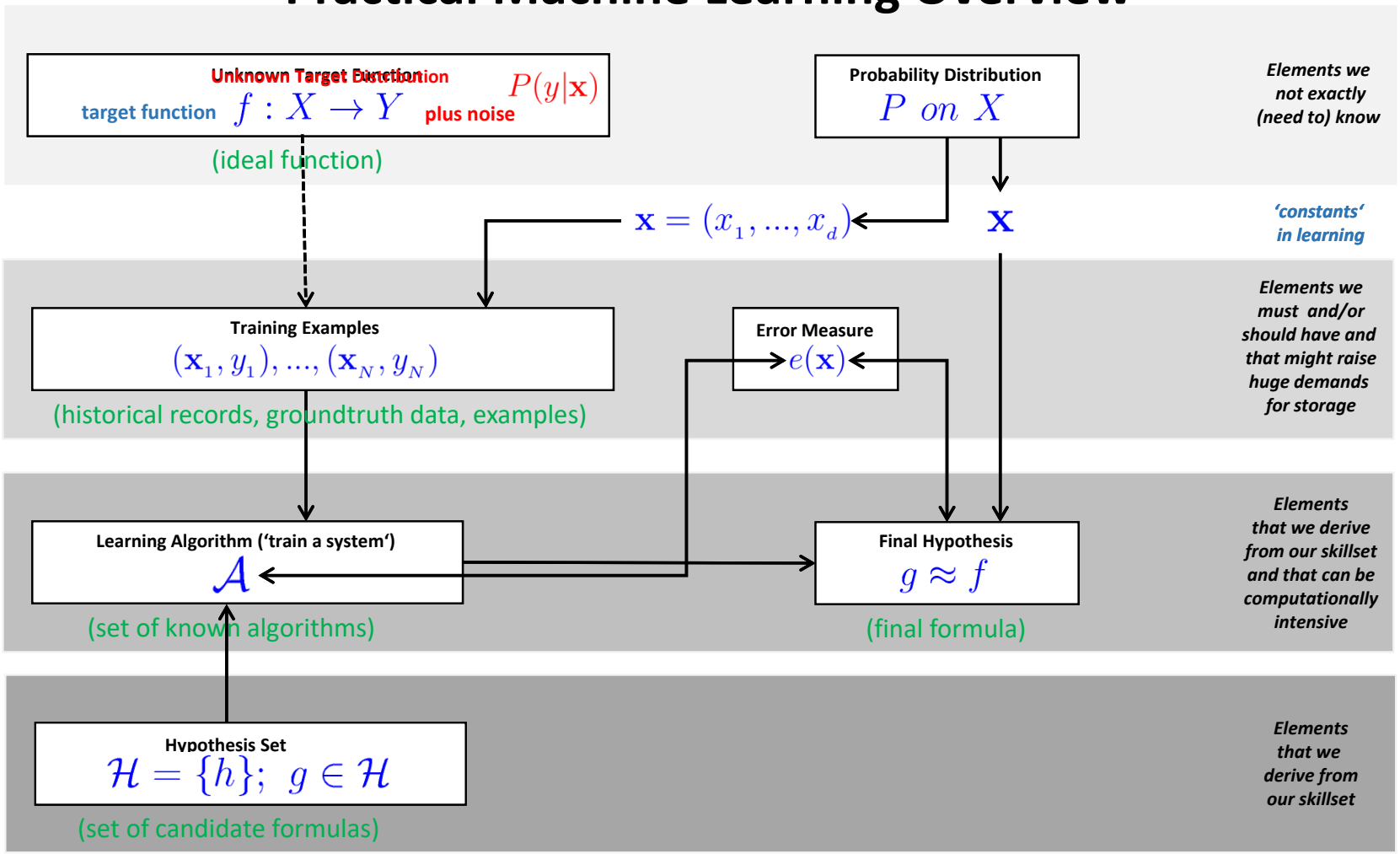
$$e(\mathbf{x})$$




```
log(1 + exp(-yw^T x))
```

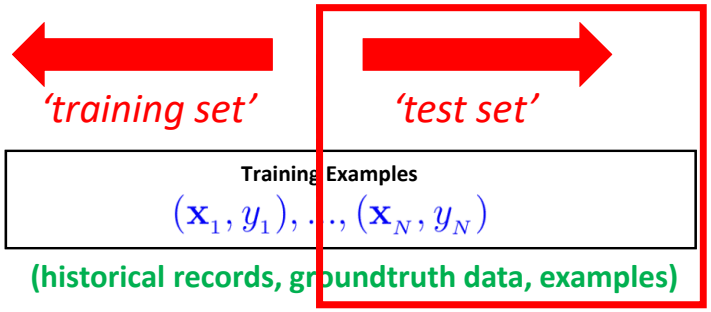
(loss function, logistic loss)

Practical Machine Learning Overview



Step 7: Logistic Regression Model Evaluation (1)

- Model Evaluation
  - The trained logistic regression model is available as ‘model‘ and used the Food_Inspections1.csv data set
  - Predict with ‘model‘ using unseen data (aka ‘test set‘) from dedicated file via the Food_Inspections2.csv (also available in MS Azure WSAB storage)
  - Original goal: Predict results of new inspections based on violations text



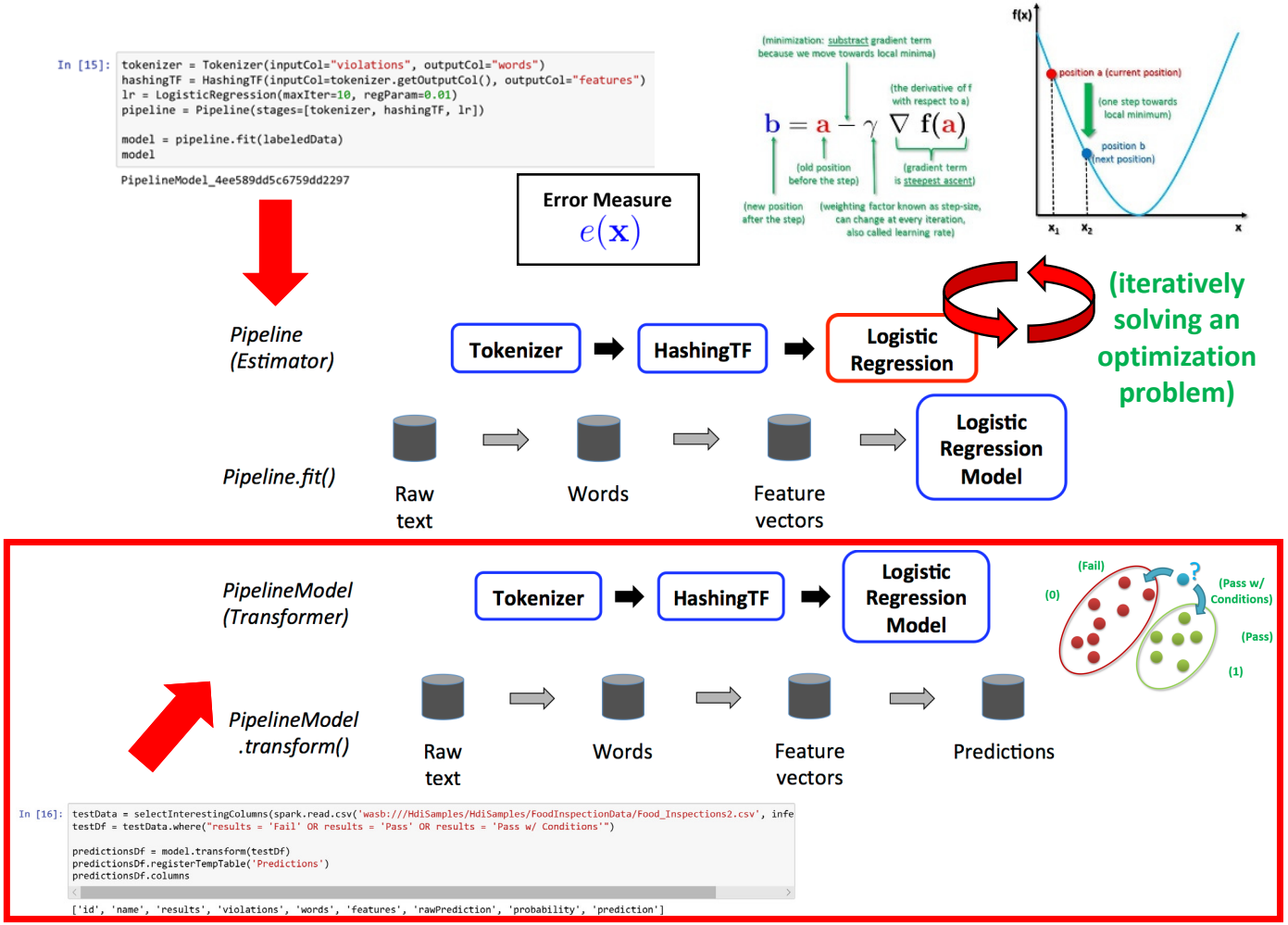

```
In [16]: testData = selectInterestingColumns(spark.read.csv('wasb:///HdiSamples/HdiSamples/FoodInspectionData/Food_Inspections2.csv', infe
         testDf = testData.where("results = 'Fail' OR results = 'Pass' OR results = 'Pass w/ Conditions'")

         predictionsDf = model.transform(testDf)
         predictionsDf.registerTempTable('Predictions')
         predictionsDf.columns

['id', 'name', 'results', 'violations', 'words', 'features', 'rawPrediction', 'probability', 'prediction']
```

- **The test dataset is different from the training dataset in order to evaluate models for unseen data**
- **Model.transform(test data frame) creates a new data frame that contains probability & predictions**

Apache Spark Pipelines & Logistic Regression Example – Testing Process

- Apache Spark Pipeline
  - Includes 'bigger picture' than just one machine learning algorithm

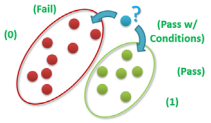
- **A data science pipeline in Apache Spark is a sequence of stages with either an 'Transformer' or an 'Estimator'**
- **The resulting PipelineModel is created as 'Transformer' and exists within memory and can be used: e.g., testing (later)**
- **When the PipelineModel is called with transform() using a test dataset (i.e., dataframe with test data that is different from the training data for evualations of the model), the test data is passed through the fitted pipeline in order**
- **Each stage will perform transform() updates on the dataset and passes it to the next stage**
- **The key benefit of data science pipelines in general and Apache Spark pipelines in particular is to ensure that the training data and the testing data go through identical feature processings steps (i.e., no need for redefine Tokenizer, etc.)**



modified from [13] Apache Spark Pipelines

Step 7: Logistic Regression Model Evaluation (2)

- Model Evaluation
  - Machine learning needs to perform good ‘out-of-sample‘
  - Goal is to get future data set elements correctly classified
  - Predicted class can be only 0.0 or 1.0
  - One example prediction with features & class
  - Using statistics to understand the model performance



In [17]:
```
predictionsDf.take(1)
```

```
[Row(id=580320, name=u"COUSIN'S GRILL", results=u'Fail', violations=u'37. TOILET ROOM DOORS SELF CLOSING: DRESSING ROOMS WITH L
OCKERS PROVIDED: COMPLETE SEPARATION FROM LIVING/SLEEPING QUARTERS - Comments: INSTRUCTED TO FIX BROKEN SELF CLOSING DEVICE AT
WASHROOM DOOR. | 40. REFRIGERATION AND METAL STEM THERMOMETERS PROVIDED AND CONSPICUOUS - Comments: All food establishments tha
t display, prepare, or store potentially hazardous foods shall have calibrated metal stem thermometers, provided and conspicuou
s, for refrigerated and hot food units.  INSTRUCTED TO PROVIDE THERMOMETTER VISIBLE INSIDE ALL COOLERS. | 36. LIGHTING: REQUIRE
D MINIMUM FOOT-CANDLES OF LIGHT PROVIDED, FIXTURES SHIELDED - Comments: Shielding to protect against broken glass falling into
food shall be provided for all artificial lighting sources in preparation, service, and display facilities.  LIGHTBULBS MUST BE
SHIELDED INSIDE HOT HOLDING DISPLAY UNIT OR PROVIDE SHATTERFROOFED LIGHTBULB.  | 18. NO EVIDENCE OF RODENT OR INSECT OUTER OPE
NINGS PROTECTED/RODENT PROOFED, A WRITTEN LOG SHALL BE MAINTAINED AVAILABLE TO THE INSPECTORS - Comments: All necessary control
measures shall be used to effectively minimize or eliminate the presence of rodents, roaches, and other vermin and insects on t
he premises of all food establishments, in food-transporting vehicles, and in vending machines.  FOUND FRONT ENTRANCE DOOR NOT
RODENT PROOFED. NOTED 1/2 INCH GAP/OPENING AT THE BOTTOM DOOR. INSTRUCTED TO RODENT-PROOFED BOTTOM OF THE SAID DOOR. SERIOUS CI
TATION ISSUED 7-38-020 | 41. PREMISES MAINTAINED FREE OF LITTER, UNNECESSARY ARTICLES, CLEANING  EQUIPMENT PROPERLY STORED - Co
mments: All parts of the food establishment and all parts of the property used in connection with the operation of the establis
hment shall be kept neat and clean and should not produce any offensive odors.  INSTRUCTED TO CLEAN AND MAINTAIN REAR OUTSIDE A
REA.', words=u'37.', u'toilet', u'room', u'doors', u'self', u'closing:', u'dressing', u'rooms', u'with', u'lockers', u'provide
d:', u'complete', u'separation', u'from', u'living/sleeping', u'quarters', u'-', u'comments:', u'instructed', u'to', u'fix',
u'broken', u'self', u'closing', u'device', u'at', u'washroom', u'door.', u'|', u'40.', u'refrigeration', u'and', u'metal', u'st
em', u'thermometers', u'provided', u'and', u'conspicuous', u'-', u'comments:', u'all', u'food', u'establishments', u'that', u'd
isplay,', u'prepare,', u'or', u'store', u'potentially', u'hazardous', u'foods', u'shall', u'have', u'calibrated', u'metal', u's
tem', u'thermometers,', u'provided', u'and', u'conspicuous,', u'for', u'refrigerated', u'and', u'hot', u'food', u'units.', u'',
u'instructed', u'to', u'provide', u'thermometter', u'visible', u'inside', u'all', u'coolers.', u'|', u'36.', u'lighting:', u're
quired', u'minimum', u'foot-candles', u'of', u'light', u'provided,', u'fixtures', u'shielded', u'-', u'comments:', u'shieldin
g', u'to', u'protect', u'against', u'broken', u'glass', u'falling', u'into', u'food', u'shall', u'be', u'provided', u'for', u'a
ll', u'artificial', u'lighting', u'sources', u'in', u'preparation,', u'service,', u'and', u'display', u'facilities.', u'', u'li
ghtbulbs', u'must', u'be', u'shielded', u'inside', u'hot', u'holding', u'display', u'unit', u'or', u'provide', u'shatterfroofe
d', u'lightbulb.', u'', u'', u'|', u'18.', u'no', u'evidence', u'of', u'rodent', u'or', u'insect', u'outer', u'openings', u'pro
tected/rodent', u'proofed,', u'a', u'written', u'log', u'shall', u'be', u'maintained', u'available', u'to', u'the', u'inspector
s', u'-', u'comments:', u'all', u'necessary', u'control', u'measures', u'shall', u'be', u'used', u'to', u'effectively', u'minim
ize', u'or', u'eliminate', u'the', u'presence', u'of', u'rodents,', u'roaches,', u'and', u'other', u'vermin', u'and', u'insect
s', u'on', u'the', u'premises', u'of', u'all', u'food', u'establishments,', u'in', u'food-transporting', u'vehicles,', u'and',
u'in', u'vending', u'machines.', u'', u'found', u'front', u'entrance', u'door', u'not', u'rodent', u'proofed.', u'noted', u'1/
2', u'inch', u'gap/opening', u'at', u'the', u'bottom', u'door.', u'instructed', u'to', u'rodent-proofed', u'bottom', u'of', u't
he', u'said', u'door.', u'serious', u'citation', u'issued', u'7-38-020', u'|', u'41.', u'premises', u'maintained', u'free', u'o
f', u'litter,', u'unnecessary', u'articles,', u'cleaning', u'', u'equipment', u'properly', u'stored', u'-', u'comments:', u'al
l', u'parts', u'of', u'the', u'food', u'establishment', u'and', u'all', u'parts', u'of', u'the', u'property', u'used', u'in',
u'connection', u'with', u'the', u'operation', u'of', u'the', u'establishment', u'shall', u'be', u'kept', u'neat', u'and', u'cle
an', u'and', u'should', u'not', u'produce', u'any', u'offensive', u'odors.', u'', u'instructed', u'to', u'clean', u'and', u'mai
ntain', u'rear', u'outside', u'area.'], features=SparseVector(262144, {1972: 1.0, 2340: 1.0, 2786: 1.0, 3168: 1.0, 4871: 1.0, 5
429: 1.0, 6857: 2.0, 7367: 1.0, 7453: 1.0, 8[illegible]0450: 1.0, 13142: 1.0, 13963: 1.0, 14114: 1.0, 14311: 5.0, 14
668: 1.0, 14898: 1.0, 15945: 1.0, 16324: 1.0, 16332: 2.0, 18504: 2.0, 20326: 1.0, 20654: 1.0, 21316: 2.0, 27116: 1.0, 27619: 1.
0, 29399: 1.0, 32756: 1.0, 33151: 1.0, 34036: 1.0, 39302: 1.0, 39926: 1.0, 40268: 1.0, 40856: 1.0, 42337: 1.0, 42343: 2.0, 4311
7: 1.0, 43583: 1.0, 45531: 5.0, 47453: 1.0, 47462: 3.0, 48448: 1.0, 48648: 2.0, 48724: 1.0, 50223: 1.0, 53651: 1.0, 56498: 1.0,
59760: 1.0, 63091: 4.0, 67005: 1.0, 67156: 1.0, 68281: 2.0, 68374: 2.0, 69094: 1.0, 74466: 1.0, 74904: 1.0, 80107: 1.0, 81932:
1.0, 83569: 1.0, 84053: 2.0, 85836: 1.0, 87239: 1.0, 88203: 1.0, 90859: 3.0, 91006: 1.0, 91677: 12.0, 92125: 1.0, 92175: 1.0, 9
6748: 1.0, 97215: 1.0, 97633: 2.0, 99346: 1.0, 99585: 1.0, 100258: 1.0, 100734: 1.0, 101169: 1.0, 102176: 1.0, 103257: 1.0, 103
382: 4.0, 103409: 1.0, 103833: 1.0, 103838: 9.0, 104448: 1.0, 105063: 2.0, 105469: 1.0, 105518: 1.0, 107299: 1.0, 108062: 1.0,
110074: 1.0, 118283: 1.0, 119835: 1.0, 120397: 1.0, 120851: 1.0, 121133: 5.0, 123069: 1.0, 124643: 1.0, 126466: 2.0, 130125: 1.
0, 135499: 1.0, 135560: 7.0, 138328: 2.0, 139098: 2.0, 143342: 1.0, 145542: 2.0, 145718: 1.0, 145838: 1.0, 152218: 1.0, 154336:
1.0, 156250: 1.0, 156568: 1.0, 157469: 2.0, 158973: 1.0, 159927: 1.0, 161061: 2.0, 162521: 1.0, 164686: 1.0, 164735: 1.0, 16649
0: 1.0, 167114: 1.0, 167152: 5.0, 168425: 1.0, 168495: 5.0, 168976: 1.0, 172516: 1.0, 174457: 1.0, 176964: 2.0, 177916: 1.0, 17
8003: 1.0, 179255: 1.0, 184351: 2.0, 185468: 1.0, 187621: 1.0, 187664: 1.0, 193224: 2.0, 194821: 1.0, 195766: 1.0, 198277: 1.0,
200400: 1.0, 205044: 7.0, 207872: 1.0, 212952: 1.0, 214333: 1.0, 216436: 1.0, 217251: 1.0, 217720: 1.0, 222453: 4.0, 223946: 1.
0, 227410: 1.0, 227469: 1.0, 229407: 1.0, 232713: 1.0, 233903: 1.0, 237761: 1.0, 238163: 1.0, 240453: 1.0, 241725: 1.0, 243322:
1.0, 245458: 1.0, 247107: 4.0, 247598: 1.0, 248091: 1.0, 249180: 7.0, 252417: 2.0, 253475: 1.0, 253757: 1.0, 254304: 1.0, 25691
8: 1.0, 257378: 1.0, 258861: 1.0, 259190: 1.0, 259409: 1.0}), rawPrediction=DenseVector([4.4305, -4.4305]), probability=DenseVe
ctor([0.9882, 0.0118]), prediction=0.0)]
```

In [18]:
```
numSuccesses = predictionsDf.where("""(prediction = 0 AND results = 'Fail') OR
                                      (prediction = 1 AND (results = 'Pass' OR
                                                           results = 'Pass w/ Conditions'))""").count()
numInspections = predictionsDf.count()

print("There were %d inspections and there were %d successful predictions" % (numInspections, numSuccesses))
print("This is a %d%% success rate" % (float(numSuccesses) / float(numInspections) * 100))
```

In [19]:
```
%%sql -q -o true_positive
SELECT count(*) AS cnt FROM Predictions WHERE prediction = 0 AND results = 'Fail'
```

In [20]:
```
%%sql -q -o false_positive
SELECT count(*) AS cnt FROM Predictions WHERE prediction = 0 AND (results = 'Pass' OR results = 'Pass w/ Conditions')
```

In [21]:
```
%%sql -q -o true_negative
SELECT count(*) AS cnt FROM Predictions WHERE prediction = 1 AND results = 'Fail'
```

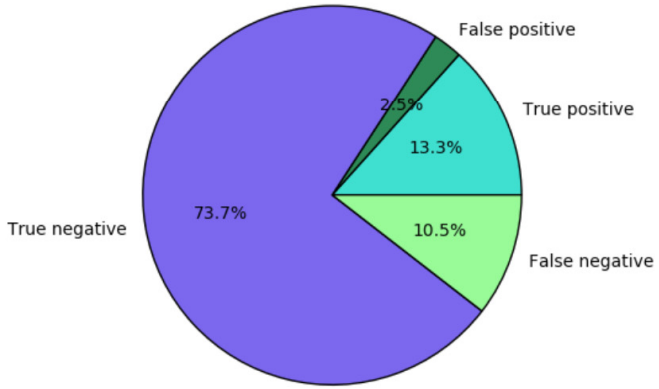
In [22]:
```
%%sql -q -o false_negative
SELECT count(*) AS cnt FROM Predictions WHERE prediction = 1 AND (results = 'Pass' OR results = 'Pass w/ Conditions')
```

- **Accurate model of relationship between violation descriptions (English) & pass/fail food inspection**

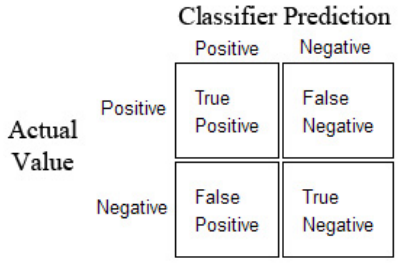
Step 7: Logistic Regression Model Evaluation (3)

- Model Evaluation
  - Visualization helps to reason about the results of using the test dataset

```
In [23]: %%local
         %matplotlib inline
         import matplotlib.pyplot as plt

         labels = ['True positive', 'False positive', 'True negative', 'False negative']
         sizes = [true_positive['cnt'], false_positive['cnt'], false_negative['cnt'], true_negative['cnt']]
         colors = ['turquoise', 'seagreen', 'mediumslateblue', 'palegreen', 'coral']
         plt.pie(sizes, labels=labels, autopct='%1.1f%%', colors=colors)
         plt.axis('equal')

Out[23]: (-1.0183947086334229, 1.0, -1.0172438621520996, 1.0000982284545898)
```

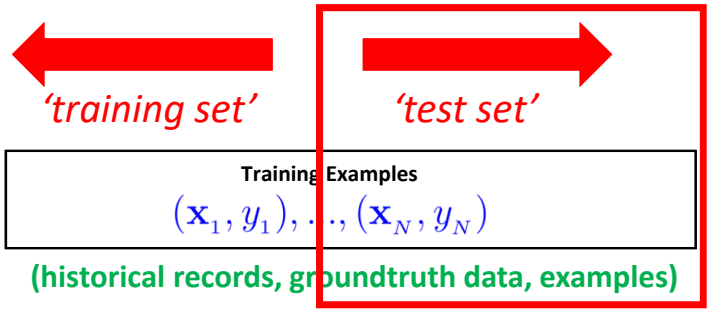


In this chart, a "positive" result refers to the failed food inspection, while a negative result refers to a passed inspection.

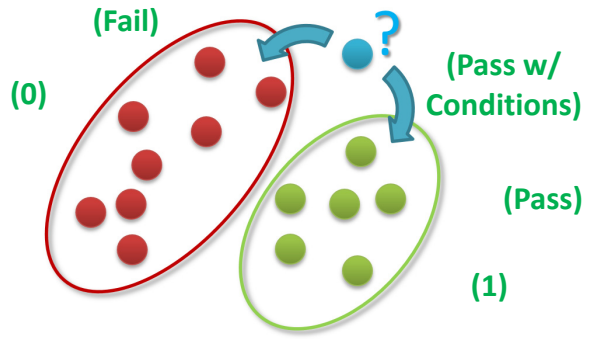


(confusion matrix)

(model evaluation for model quality checks)

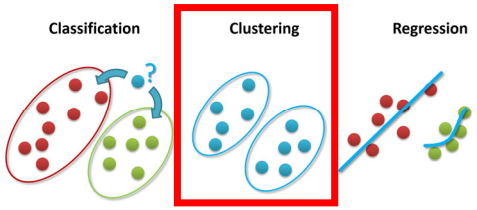


(historical records, groundtruth data, examples)

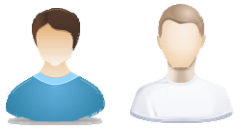


Other Mllib Algorithms – Partly Overlap with Data Mining Techniques

- Example: K-Means Clustering



[12] YouTube video, Visualization of K-Means Clustering



Customers	
ID	Name
1	Mike Jones
2	Steve Thomas
3	Peter Sloan
4	Vish Macnhi
5	Tim Albright

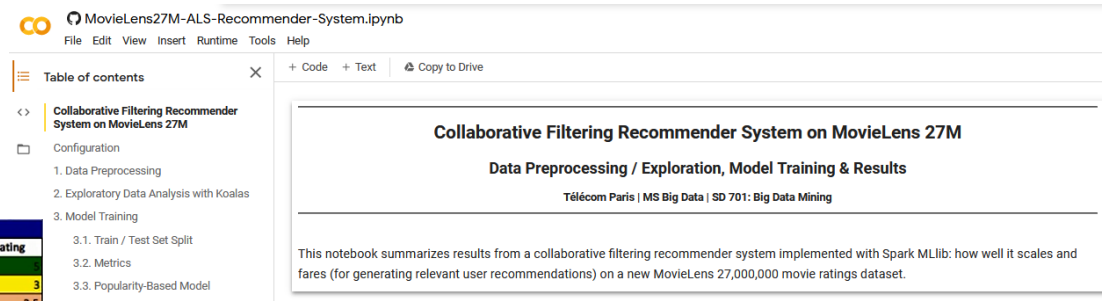
LEGEND
1
2
3
4
5

Movies		
ID	Name	Category
101	The Internship	Comedy
102	The Purge	Thriller
103	Much About Nothing	Comedy
104	Dirty Wars	Drama
105	Wish You Were Here	Drama
106	Syrup	Comedy
107	Fast & Furios	Thriller

Transactions		
Customer ID	Movie ID	Rating
1	101	5
1	102	3
1	103	2.5
2	101	2
2	102	2.5
2	103	5
2	104	2
3	101	2.5
3	104	4
3	105	4.5
3	107	5
4	101	5
4	103	3
4	104	4.5
4	106	4
5	101	4
5	102	3
5	103	2
5	104	4
5	105	3.5
5	106	4

- Example: Recommendation Engine
  - Using Collaborative Filtering via Google 'Colab'

[2] Google Colaboratory



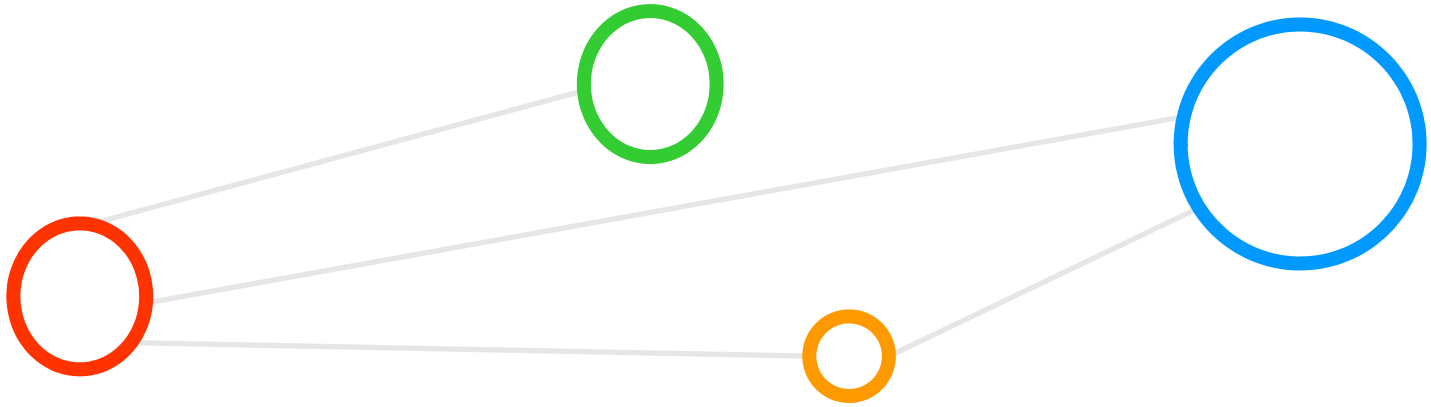
item	User 1	User 2	User 3	User 4	User 5
101	5	2	2.5	5	4
102	3	2.5			3
103	2.5	5		3	2
104		2	4.5	4.5	4
105			4.5		3.5
106				4	4
107			5		

(similar personality?)

[14] Collaborative Filtering Recommender System on Colab

➢ **Lecture 11 provides more details on using recommender engines & that are partly considered as data mining technique**

Lecture Bibliography



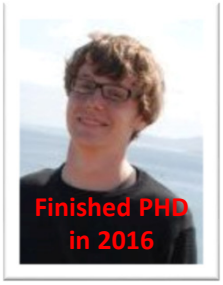
Lecture Bibliography (1)

- [1] Apache Spark Web page, Online: http://spark.apache.org/
- [2] Google Colaboratory, Online: https://colab.research.google.com
- [3] Amazon Web Services Web Page, Online: https://aws.amazon.com
- [4] Microsoft Azure HDInsight Service, Online: https://azure.microsoft.com/en-us/services/hdinsight/
- [5] www.big-data.tips, 'Gradient Descent', Online: http://www.big-data.tips/gradient-descent
- [6] Understanding Parallelization of Machine Learning Algorithms in Apache Spark, Online: https://www.slideshare.net/databricks/understanding-parallelization-of-machine-learning-algorithms-in-apache-spark/11
- [7] Species Iris Group of North America Database, Online: http://www.signa.org
- [8] Project Jupyter, Online: https://jupyter.org/
- [9] Microsoft Azure Portal Hub, Online: https://portal.azure.com/#home
- [10] MS Azure HDInsight Quickstart: Create a Spark cluster in HDInsight using template, Online: https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-jupyter-spark-sql
- [11] MS Azure Storage, Online: https://azure.microsoft.com/en-us/pricing/details/storage/

Lecture Bibliography (2)

- [12] YouTube Video, ‘Visualization of k-means clustering‘, Online: https://www.youtube.com/watch?v=nXY6PxAaOk0
- [13] Apache Spark Pipelines, Online: https://spark.apache.org/docs/2.0.2/ml-pipeline.html
- [14] Collaborative Filtering Recommender System on Colab, Online: https://colab.research.google.com/github/SJD1882/Big-Data-Recommender-Systems/blob/master/notebooks/MovieLens27M-ALS-Recommender-System.ipynb

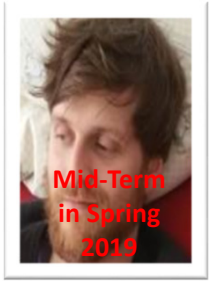
Acknowledgements – High Productivity Data Processing Research Group



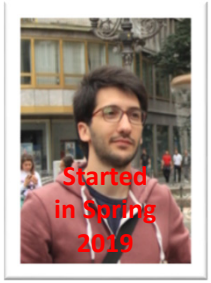
PD Dr.
G. Cavallaro

Senior PhD
Student A.S. Memon

Senior PhD
Student M.S. Memon



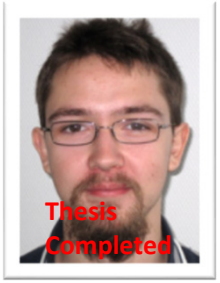
PhD Student
E. Erlingsson



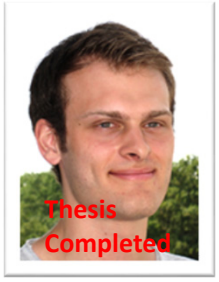
PhD Student
S. Bakarat

PhD Student
R. Sedona

Dr. M. Goetz
(now KIT)



MSc M.
Richerzhagen
(now other division)



MSc
P. Glock
(now INM-1)

MSc
C. Bodenstein
(now
Soccerwatch.tv)

MSc Student
G.S. Guðmundsson
(Landsverkjun)

This research group has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 763558 (DEEP-EST EU Project)


Data
Science
Big
Research
science
analysis
methods
big
simulations
processing
technologies
TB
computing
European
Scientific
computational
research
Computational
E.g
Centre
climate
Computing
nodes
centers
storage
ENES
EUDAT
Supercomputing
images
performance
datasets
Juelich
HPC
Climate
modelling
structures
access
databases
DLCL
Service
data-intensive
distributed
large-scale
management
community
Environment
Infrastructure
important
Enable
Modelling
services
resources
year
new
Project
PB
High
Hardware
System
DLCLs
increasing
brain
EU
often
system
compute
hundreds
Landscape
Services
network
General
Structure
project
directory
Earth
Health
day
Technologies
AR5
Simulation
systems
Energy
NASA
Germany
Provide
Understanding
acquisition
e.g
JSC
use
codes
SMAQ
international
model
engineering
References
formats
Cross-Disciplinary
like
forms
cross-disciplinary
measurement
Policy-based
within
fusion
funding
concepts
device
Analysis
Computer
expertise
disciplines
results
Approaches
construction
Fusion
Key
scientific
basis
files
different
DKRZ
Use
etc
via
physical
IPCC
Common
Methods
Using
Work
external
levels
Matters
Concrete
Summary
Large
analyze
stored
Resources
GB
MapReduce
create
Level
Test
cores
current
Nutshell
Processing