PREDICT NATIVE ADS: HANDS-ON MACHINE LEARNING EXERCISE USING SPARK.ML

ALEXEY SVYATKOVSKIY

OUTLINE

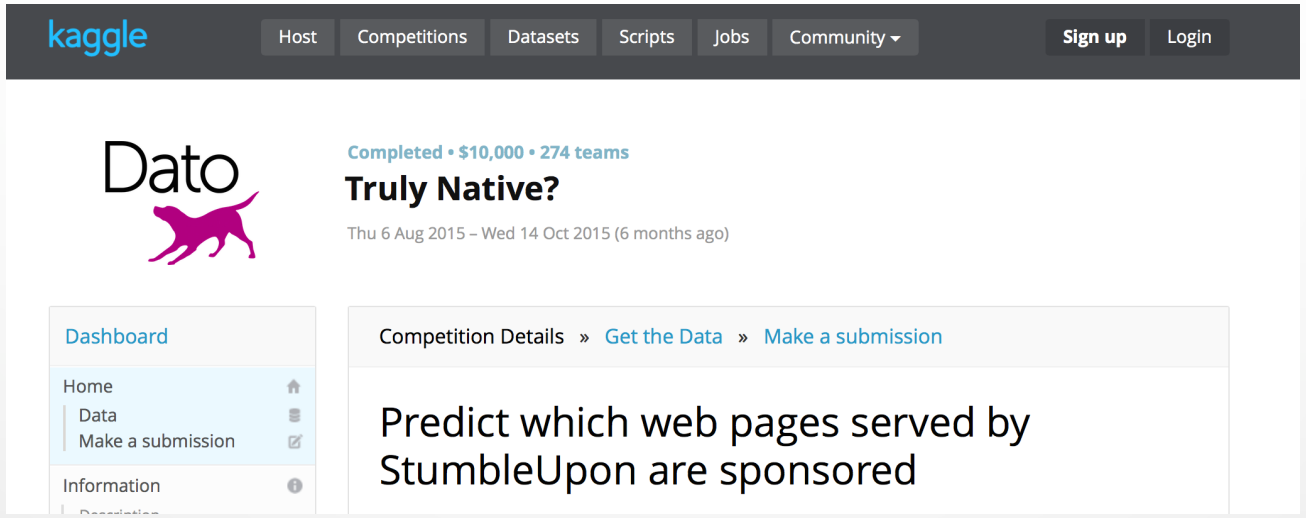
- PROBLEM DESCRIPTION
- DATA PREPROCESSING: WEB (SCREEN) SCRAPING
- CHOICE OF TOOLS: SPARK.ML, DATAFRAMES
- FEATURE ENGINEERING
- BUILDING A CLASSIFIER
  - DECISION TREES, ENSEMBLES: BOOTSTRAP AGGREGATION, BOOSTING
  - SUPPORT VECTOR MACHINES
  - REGULARIZATION, OVERFITTING AND CROSS VALIDATION
- SPARK.ML PIPELINES

PREREQUISITES: GETTING STARTED

- THESE SLIDES GIVE AN OVERVIEW OF CONCEPTS USED IN THE STUDY, ALL THE DETAILS AND THE ANALYSIS CODE ARE PROVIDED IN THE REPOSITORY
- LINK TO THE MEETUP REPOSITORY: HTTPS://GITHUB.COM/ASVYATKOVSKIY/MLMEETUP2016

PROBLEM DESCRIPTION

- THE DESCRIPTION OF THE PROBLEM CAN BE FOUND HERE: [HTTPS://WWW.KAGGLE.COM/C/DATO-NATIVE](https://www.kaggle.com/c/dato-native)



- THE TASK IS TO PREDICT WHICH WEB PAGES SERVED BY STUMBLEUPON COMPANY ARE SPONSORED.
- WHEN NATIVE ADVERTISING IS DONE RIGHT, USERS AREN'T DESPERATELY SCANNING AN AD FOR A HIDDEN "X". IN FACT, THEY DON'T EVEN KNOW THEY'RE VIEWING ONE. TO PULL THIS OFF, NATIVE ADS NEED TO BE JUST AS INTERESTING, FUN, AND INFORMATIVE AS THE UNPAID CONTENT ON A SITE.
- IF MEDIA COMPANIES CAN BETTER IDENTIFY POORLY DESIGNED NATIVE ADS, THEY CAN KEEP THEM OFF YOUR FEED AND OUT OF YOUR USER EXPERIENCE.

DATA PREPROCESSING

- IN OUR TASK, DATA COMES AS A SET OF UNSTRUCTURED HTML DOCUMENTS
- WE ARE GOING TO USE CSS SELECTORS AND BeautifulSoup TO PARSE THE HTML DOM STRUCTURE AND EXTRACT AND STRUCTURE THE INFO CONTAINED THERE (WEB SCRAPING, OR SCREEN SCRAPING)
- THE OUTPUT IS STORED AS A JSON AND CONVERTED TO AVRO FOR FURTHER USE

Data

```
{"namespace": "html.avro",
     "type": "record",
     "name": "Html",
     "fields": [
         {"name": "id", "type": "string"},
         {"name": "images", "type": {"type":"array",
"items":"string"}},
         {"name": "links", "type": {"type":"array",
"items":"string"}},
         {"name": "text", "type": "string"},
         {"name": "title", "type": {"type":"array",
"items":"string"}}
     ]
    }
```

Labels

```
{"namespace": "html.avro",
     "type": "record",
     "name": "Html",
     "fields": [
         {"name": "id", "type": "string"},
         {"name": "label", "type": "double"}
     ]
    }
```

Foreign key to join

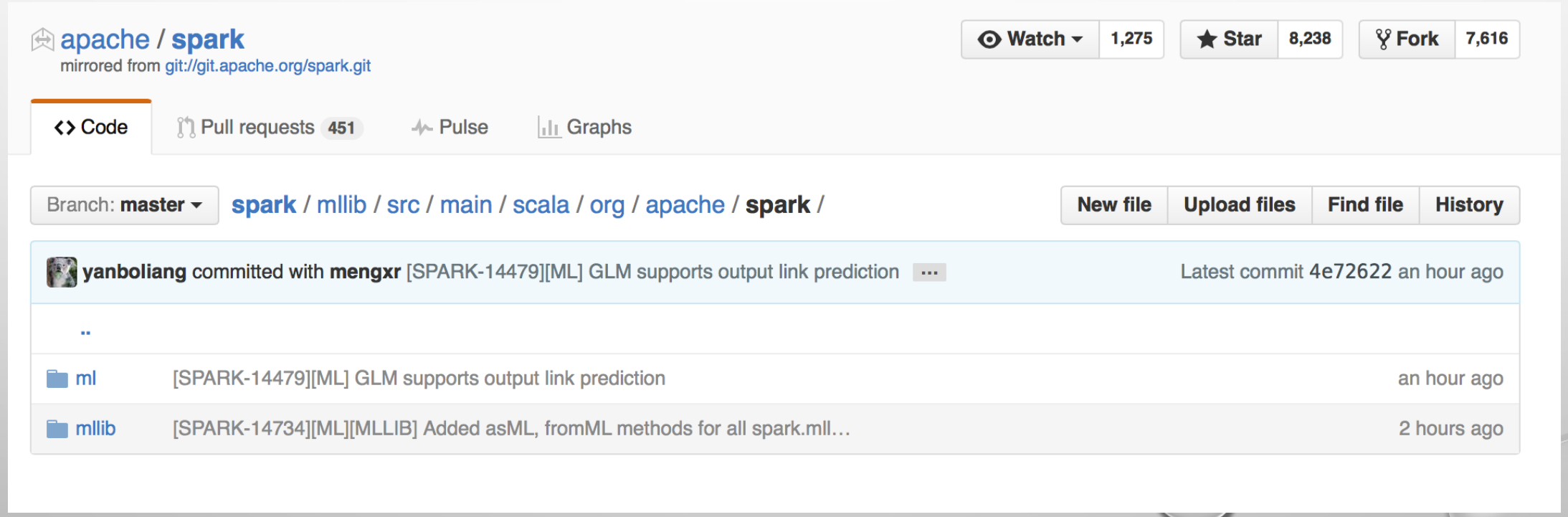
WHY DATAFRAMES?

- SPARK SQL AND ITS DATAFRAMES ARE ESSENTIAL FOR SPARK PERFORMANCE WITH MORE EFFICIENT STORAGE OPTIONS, ADVANCED OPTIMIZER, AND DIRECT OPERATIONS ON SERIALIZED DATA.
  - INTRODUCED IN SPARK 1.3 (USED TO BE CALLED SCHEMA RDD AND INHERITED DIRECTLY FROM RDD)
- LIKE RDDS, DATAFRAMES REPRESENT DISTRIBUTED COLLECTIONS, WITH ADDITIONAL SCHEMA INFORMATION NOT FOUND IN RDDS
  - THIS ADDITIONAL SCHEMA INFORMATION IS USED TO PROVIDE A MORE EFFICIENT STORAGE LAYER AND IN THE OPTIMIZER
- COMPARED TO WORKING WITH RDDS, DATAFRAMES ALLOW SPARK'S OPTIMIZER TO BETTER UNDERSTAND OUR CODE AND OUR DATA

Based on "High performance Spark" by H. Karau et al and personal experience

SPARK ML (I)

- THE SPARK.ML PACKAGE AIMS TO PROVIDE A UNIFORM SET OF HIGH-LEVEL APIS BUILT ON TOP OF DATAFRAMES ALLOWING USERS TO CREATE MACHINE LEARNING PIPELINES
  - THE PIPELINE CONCEPT IS INSPIRED BY THE SCIKIT-LEARN PROJECT



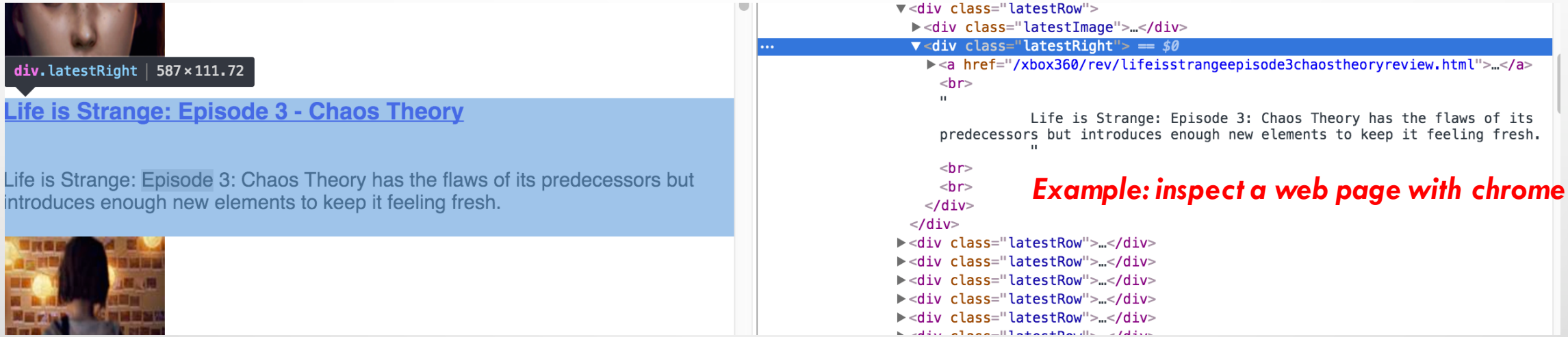
SPARK ML (II)

- SPARK ML STANDARDIZES APIs FOR MACHINE LEARNING ALGORITHMS TO MAKE IT EASIER TO COMBINE MULTIPLE ALGORITHMS INTO A SINGLE PIPELINE OR WORKFLOW
- HERE ARE THE BASIC COMPONENTS OF A PIPELINE

- **TRANSFORMER**: A TRANSFORMER IS AN ALGORITHM WHICH CAN TRANSFORM ONE DATAFRAME INTO ANOTHER DATAFRAME. EACH TRANSFORMER CLASS HAS A Transform METHOD
  - E. G. TOKENIZER, STOP WORD REMOVER ARE TRANSFORMERS
- **ESTIMATOR**: AN ESTIMATOR IS AN ALGORITHM WHICH CAN BE FIT ON A DATAFRAME TO PRODUCE A TRANSFORMER. E.G., A LEARNING ALGORITHM IS AN ESTIMATOR WHICH TRAINS ON A DATAFRAME AND PRODUCES A MODEL. EACH ESTIMATOR CLASS HAS A Fit METHOD
  - E.G. RANDOMFOREST CLASSIFIER
- **PARAMETER**: ALL TRANSFORMERS AND ESTIMATORS SHARE A COMMON API FOR SPECIFYING PARAMETERS

FEATURE ENGINEERING

- WHAT KIND OF INFORMATION CONTAINED IN THE HTML DOCUMENT CAN BE USED TO CONSTRUCT FEATURES FOR CLASSIFICATION?
  - MOST OF THE WEB PAGE CONTENT IS TEXT: TEXT FEATURES (TF-IDF, BAG OF WORDS ETC)



  - IN ADDITION THERE IS HYPERLINKS, IMAGES: WE WILL CALL THEM ADHOC FEATURES

- FOR THE TEXT FEATURES USE BAG-OF-WORDS APPROACH, TF/IDF, N-GRAMS, STEMMING, PART OF SPEECH TAGGING, FEATURE HASHING
- START WITH COUNT FEATURES FOR THE HYPERLINK AND IMAGE INFORMATION ON THE WEB PAGE

EXTRACT TEXT FEATURES (I)

- **TOKENIZE**: A SIMPLE WAY TO TOKENIZE A STRING (SENTENCE) IN PYTHON IS USING THE SPLIT() METHOD
  - SPARK ML HAS 2 TOKENIZERS: THE (DEFAULT) Tokenizer, WHICH TOKENIZES ON WHITESPACE AND RegexTokenizer WHICH ALLOWS TO SPECIFY A CUSTOM PATTERN TO TOKENIZE ON OTHER THAN WHITE SPACE. THE LATTER IS MORE FLEXIBLE
- **N-GRAMS**: INSTEAD OF LOOKING AT JUST SINGLE WORDS, IT IS ALSO USEFUL TO LOOK AT N-GRAMS, THE N-WORD LONG SEQUENCES
- **REMOVE STOPWORDS**: OMIT CERTAIN COMMON WORDS WHICH BEAR NO MEANING OF DOCUMENTS LIKE "A", "AN", AND "THE"
  - SPARK ML CONTAINS A STANDARD LIST OF SUCH STOP WORDS FOR ENGLISH. ONE CAN INCLUDE ANY CUSTOM STOPWORDS, IF NEED BE
- **STEMMING**: IT WOULD HAVE BEEN USEFUL TO IDENTIFY WORDS LIKE "COMPUTER" AND "COMPUTERS" AS ONE WORD. THE PROCESS OF REPLACING THEM BY A COMMON ROOT, OR **STEM**, IS CALLED STEMMING - THE STEM WILL NOT, IN GENERAL, BE A FULL WORD ITSELF

EXTRACT TEXT FEATURES (II)

- **FEATURE HASHING**: IS A WAY OF VECTORIZING FEATURES, E.G. TURNING TEXT FEATURES INTO INDICES IN A VECTOR OR MATRIX. IT WORKS BY APPLYING A HASH FUNCTION TO THE FEATURES AND USING THEIR HASH VALUES AS INDICES
  - A HASH FUNCTION TAKES A STRING AS AN INPUT AND SPITS OUT A NUMBER, WITH THE DESIRED PROPERTY BEING THAT DIFFERENT INPUTS USUALLY PRODUCE DIFFERENT OUTPUTS (NO HASH COLLISIONS)
- **TF-IDF WEIGHTING**: (TERM-FREQUENCY INVERSE DOCUMENT FREQUENCY) IS A FEATURE VECTORIZATION METHOD IN TEXT MINING TO REFLECT IMPORTANCE OF A TERM *t* TO A DOCUMENT *d* IN CORPUS *D*

$$IDF(t, D) = \log \frac{D + 1}{DF(t, D) + 1}$$

$$TFIDF(t, d, D) = TF(t, d) \bullet IDF(t, D)$$

SUPERVISED MACHINE LEARNING

- LET $X = \{X_{ji}\}$ BE THE NXP FEATURE MATRIX
- AND y_j BE A N-VECTOR OF LABELS
- **SUPERVISED LEARNING** IS THE MACHINE LEARNING TASK OF INFERRING A FUNCTION

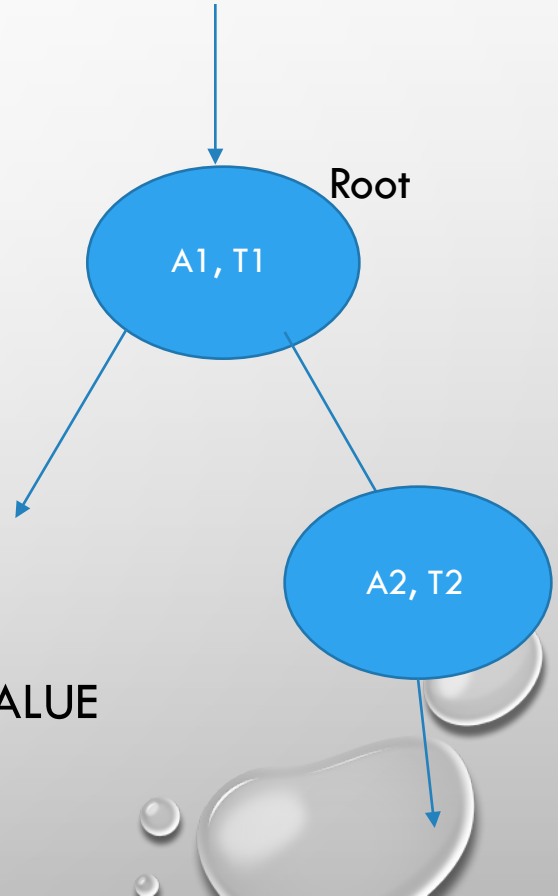
$$f(X_{j\bullet}) = y_j$$

FROM *LABELED* TRAINING DATA

- DECISION TREES, RANDOM FORESTS AND SUPPORT VECTOR MACHINES ARE SOME COMMONLY USED SUPERVISED MACHINE LEARNING TECHNIQUES

DECISION TREES

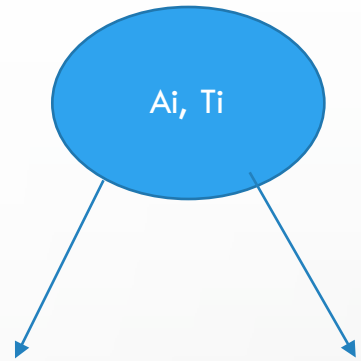
- A DECISION TREE IS A BINARY TREE. AT EACH OF THE INTERNAL NODES, IT CHOOSES A FEATURE *i* AND A THRESHOLD *T*
  - EACH LEAF HAS A VALUE
- EVALUATION OF THE MODEL IS JUST A TRAVERSAL OF THE TREE FROM THE ROOT
- AT EACH NODE, FOR EXAMPLE *j*, WE GO DOWN THE LEFT BRANCH IF $X_{ji} < T$ AND THE RIGHT BRANCH OTHERWISE
- THE VALUE OF THE MODEL $f(X_{ji})$ IS THE VALUE AT THE VALUE AT THE TERMINATING LEAF OF THIS TRAVERSAL
- **CLASSIFICATION AND REGRESSION TREE (CART)** ANALYSIS IS AN UMBRELLA TERM USED TO REFER TO THE DECISION TREES WHICH OUTPUT THE CLASS LABEL OR A REAL VALUE



DECISION TREES: ID3

{Ai, Ti} – set of attributes and thresholds to choose from

Initial training dataset *S having X classes*



- THE ID3 ALGORITHM BEGINS WITH THE ORIGINAL SET S AT THE ROOT NODE
- IT ITERATES THROUGH EVERY UNUSED ATTRIBUTE OF THE SET AND CALCULATES THE ENTROPY (OR INFORMATION GAIN) OF THAT ATTRIBUTE
  - IT THEN SELECTS THE ATTRIBUTE WHICH HAS THE LARGEST INFORMATION GAIN
  - THE SET IS THEN SPLIT BY THE SELECTED ATTRIBUTE TO PRODUCE SUBSETS OF THE DATA
- THE ALGORITHM CONTINUES TO RECURSE ON EACH SUBSET, CONSIDERING ONLY ATTRIBUTES NEVER SELECTED BEFORE
- RECURSION ON A SUBSET MAY STOP IN ONE OF THESE CASES:
  - EVERY ELEMENT IN THE SUBSET BELONGS TO THE SAME CLASS. THEN THE NODE IS TURNED INTO A LEAF AND LABELLED WITH THE CLASS OF THE EXAMPLES
  - THERE ARE NO MORE ATTRIBUTES TO BE SELECTED, BUT THE EXAMPLES STILL DO NOT BELONG TO THE SAME CLASS, THEN THE NODE IS TURNED INTO A LEAF AND LABELLED WITH THE MOST COMMON CLASS OF THE EXAMPLES IN THE SUBSET
  - THERE ARE NO EXAMPLES IN THE SUBSET

$p(x)$ - fraction of elements of class x

Entropy of the set *S* would be:

$$H(S) = -\sum_{x \in X} p(x) \log_2 p(x)$$

Information gain for a given attribute *A* on the set *S*:

$$IG(A, S) = H(S) - \sum_{t \in T} p(t) H(t)$$

RANDOM FOREST

- A **RANDOM FOREST** IS JUST AN ENSEMBLE OF DECISION TREES
- THE PREDICTED VALUE IS JUST THE AVERAGE OF THE TREES (FOR BOTH REGRESSION AND CLASSIFICATION PROBLEMS - FOR CLASSIFICATION PROBLEMS, IT IS THE PROBABILITIES THAT ARE AVERAGED).
- WHY RANDOM FORESTS? THERE IS TWO SOURCES OF RANDOMNESS:
  - **BOOTSTRAP AGGREGATION (SUBSAMPLING)**: EACH TREE IS TRAINED ON A SUBSET OF DATA SELECTED AT RANDOM WITH REPLACEMENT
  - **SELECT SUBSET OF TRAINING FEATURES**: THE OPTIMAL SPLIT COMES FROM A RANDOMLY SELECTED SUBSET OF THE FEATURES
- **EXTREMELY RANDOM FORESTS**: INSTEAD OF CHOOSING THE OPTIMAL SPLIT AMONGST A SUBSET OF FEATURES, WE CHOOSE RANDOM VALUES AMONGST RANDOMLY GENERATED THRESHOLDS
- *DETAILS ON THE RandomForestClassifier IN THE SPARK.ML ARE IN THE NOTEBOOK*

BOOSTING: EPSILON BOOST

- BOOSTING IS AN ITERATIVE ALGORITHM TO REDUCE THE VARIANCE OF ENSEMBLE OF DECISION TREES (CAN BE APPLIED TO OTHER CLASSIFIERS AS WELL)
  - DECISION TREES ARE HIGH VARIANCE CLASSIFIERS
  - REWEIGHT MISCLASSIFIED EVENTS, REPEAT THE TRAINING ON THE WHOLE SAMPLE
- THE ALGORITHM:

- Initialize event weights: $W_i = \frac{1}{N}$

y_i – class labels

- Define index function: $T_m(x_i)$, +1 if the result of classification is correct, -1 otherwise
- Define loss function as $Err_m = \sum_{T_m(xi) \neq yi} W_i$ (Sum of weights for misclassified events for each tree m)
- Calculate score for each tree as: $B_m = \text{A}\bullet\log(\frac{1-Err_m}{Err_m})$
- Boost (or increase) weights $W_i \rightarrow W_i e^{B_m}$
- Renormalize all events W_i - $\rightarrow W_i \frac{}{\sum W_i}$
- Score by summing over trees, stop iteration once desired accuracy is reached

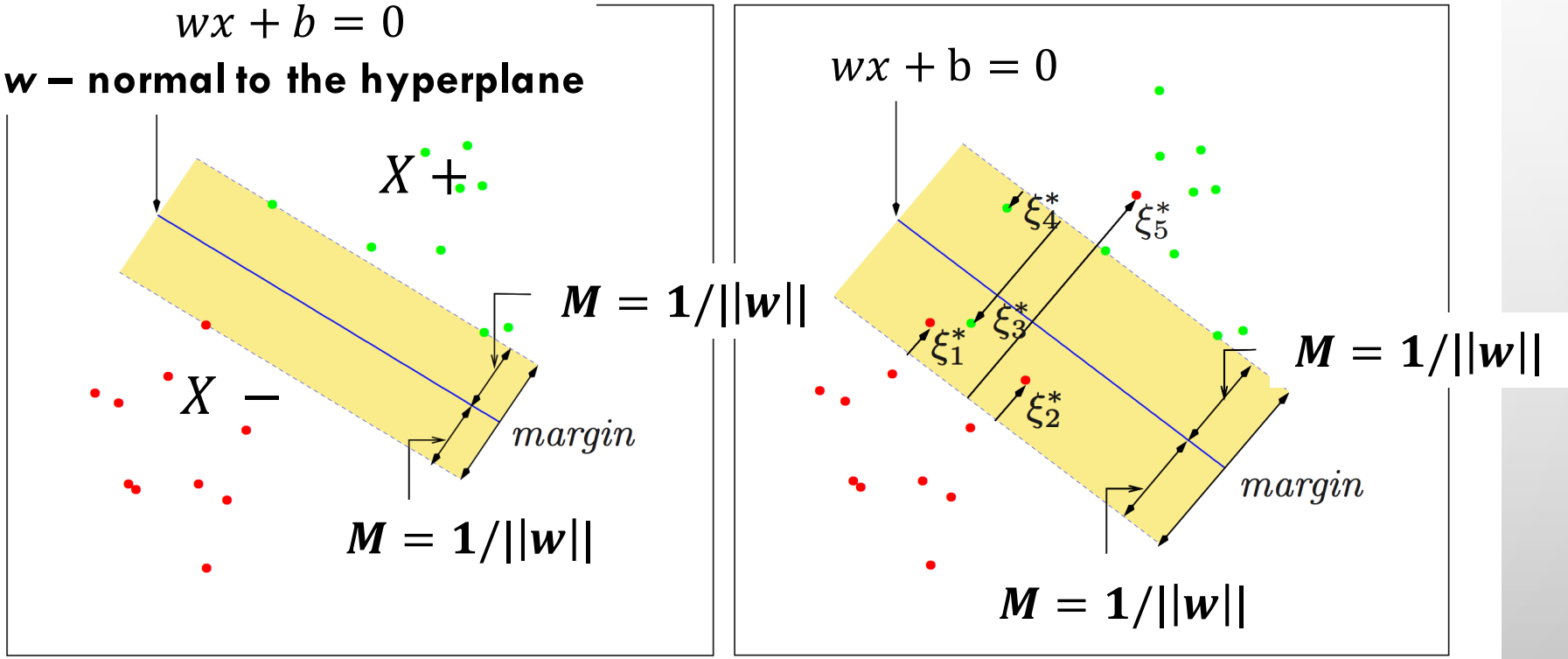
LINEAR SUPPORT VECTOR MACHINE

- **LINEAR SVM:** ASSUME WE HAVE TWO CLASSES LABELED +1 AND -1. WE ARE TRYING TO FIND A LINE OR A HYPERPLANE MAXIMALLY SEPARATING THE POINTS IN OUR TWO CLASSES

Points on the separating plane satisfy:

$$wx + b = 0$$

***w* – normal to the hyperplane**



Linearly separable case

Linearly non-separable case

LINEAR SUPPORT VECTOR MACHINE

- FOR THE LINEARLY SEPARABLE CASE, A CONDITION OF MAXIMIZING THE MARGIN BETWEEN TWO CLASSES CAN BE WRITTEN DOWN AS:

$$wx_i + b > +1, \text{for } y_i = +1,$$
$$wx_i + b < -1, \text{for } y_i = -1$$

- OR COMBINED:

$$y_i(wx_i + b) - 1 > 0, \text{ for any } y_i$$

As of now, SVM is not available in Spark.ML, but is available in MLlib

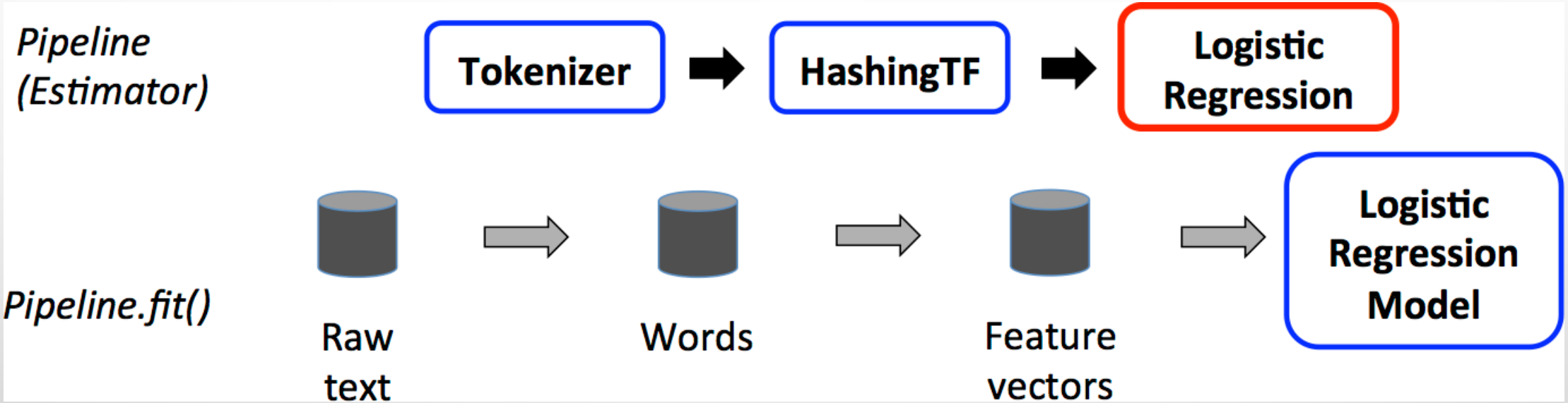
- CALCULATING THE PERPENDICULAR DISTANCE FROM THE ORIGIN FOR EACH PLANE, WE GET: $1/||w||$
- THUS ONE CAN FIND THE PAIR OF HYPERPLANES WHICH GIVES THE MAXIMUM MARGIN BY MINIMIZING $||w||^2$ SUBJECT TO THE ABOVE CONSTRAINTS. THIS IS TYPICALLY SOLVED AS A QUADRATIC PROGRAMMING PROBLEM
- IN NON SEPARABLE CASE (RIGHT PLOT), THE OBJECTIVE OF MINIMIZATION WOULD BE: $||w||^2 + C\sum_i \xi$

$$y_i(wx_i + b) > 1 - \xi_i, \text{ for any } y_i$$

- **NON-LINEAR SVM:** *FIRST MAP DATA INTO A RICHER FEATURE SPACE INCLUDING NONLINEAR FEATURES, THEN CONSTRUCT A HYPERPLANE IN THAT SPACE SO ALL OTHER EQUATIONS ARE THE SAME!*

PUTTING IT ALL INTO PIPELINE

- IN MACHINE LEARNING, IT IS COMMON TO RUN A SEQUENCE OF ALGORITHMS TO PROCESS AND LEARN FROM DATA. E.G., A SIMPLE TEXT DOCUMENT PROCESSING WORKFLOW MIGHT INCLUDE SEVERAL STAGES



- A PIPELINE IS A SEQUENCE OF STAGES, AND EACH STAGE IS EITHER A **TRANSFORMER** OR AN **ESTIMATOR**.
- THESE STAGES ARE RUN IN ORDER, AND THE INPUT DATAFRAME IS TRANSFORMED AS IT PASSES THROUGH EACH STAGE

EVALUATING A MODEL (I)

- THERE ARE IS A NUMBER OF METRICS FOR CLASSIFICATION AND THEY DEPEND ON WHETHER THE PREDICTIONS ARE GIVEN IN TERMS OF THE POTENTIAL LABEL CLASSES OR PROBABILITIES.
- RECALL THIS WELL-KNOWN TABLE

	Observation Positive	**Observation Negative**
Prediction Positive	True Positive	False Positive (Type I)
Prediction Negative	False Negative (Type II)	True Negative

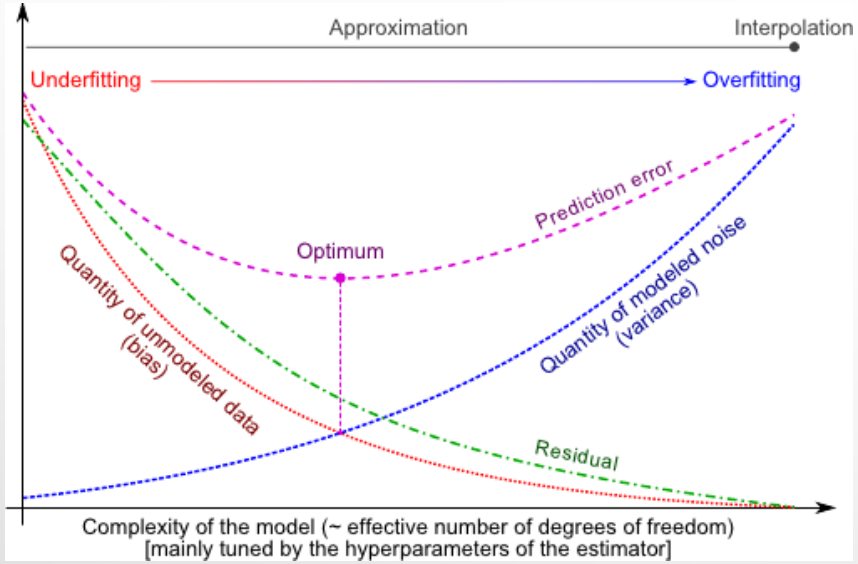
- THERE ARE MANY SUMMARY STATISTICS ONE CAN COMPUTE FROM THIS TABLE:
  - THE **ACCURACY** GIVES THE FRACTION LABELS CORRECTLY PREDICTED (TRUE POSITIVES AND TRUE NEGATIVES OVER EVERYTHING).
  - THE **HAMMING LOSS** GIVES THE FRACTION OF LABELS INCORRECTLY PREDICTED. IT IS 1 – ACCURACY.
  - THE **PRECISION** IS TRUE POSITIVES DIVIDED BY ALL POSITIVE PREDICTIONS
  - THE **RECALL** IS TRUE POSITIVES DIVIDED BY ALL POSITIVE OBSERVATIONS.
  - THERE IS ALSO **F-BETA** SCORE WHICH GIVES A WEIGHTED GEOMETRIC AVERAGE BETWEEN THE PRECISION AND RECALL
  - THE **JACCARD SIMILARITY COEFFICIENT** IS THE TRUE POSITIVES DIVIDED BY THE SUM OF TRUE POSITIVES, FALSE NEGATIVES, AND FALSE POSITIVES.

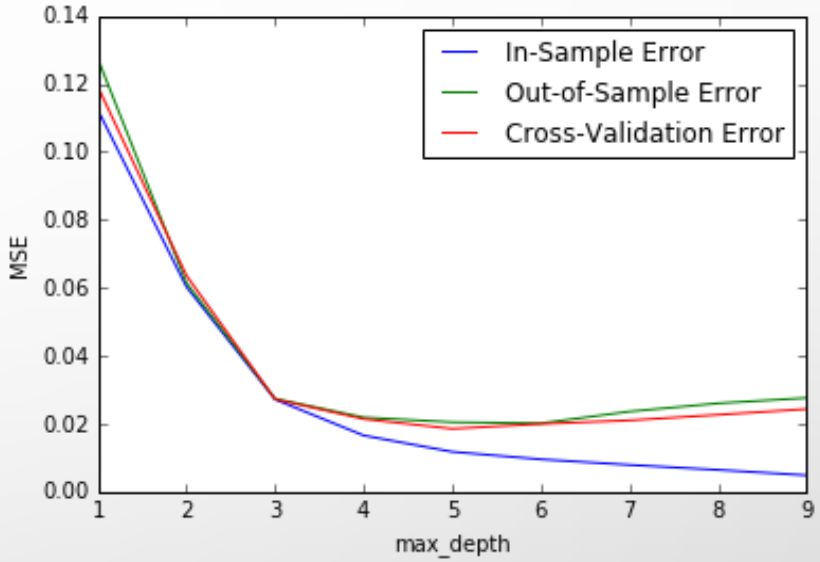
EVALUATING A MODEL (II)

- IN SPARK ML, WE HAVE TWO MAIN CLASSES FOR EVALUATORS:
- **BinaryClassificationEvaluator**: EVALUATOR FOR BINARY CLASSIFICATION, WHICH EXPECTS TWO INPUT COLUMNS: *SCORE* AND *LABEL*. IT USES AreaUnderROC (AREA UNDER THE RECEIVER OPERATING CHARACTERISTIC CURVE) AND AreaUnderPR (AREA UNDER THE PRECISION-RECALL CURVE) METRICS TO EVALUATE A CLASSIFIER
- **MulticlassClassificationEvaluator**: EVALUATOR FOR MULTI-CLASS CLASSIFICATION. EVALUATION METRICS OPTIONS ARE F1, PRECISION, RECALL

OVERFITTING AND CROSS VALIDATION

- OVERFITTING OCCURS WHEN A STATISTICAL MODEL DESCRIBES RANDOM ERROR OR NOISE INSTEAD OF THE UNDERLYING RELATIONSHIP. OVERFITTING GENERALLY OCCURS WHEN A MODEL IS EXCESSIVELY COMPLEX, SUCH AS HAVING TOO MANY PARAMETERS RELATIVE TO THE NUMBER OF OBSERVATIONS.





- CROSS VALIDATION STRATEGY: ONE COULD SPLIT THE DATA INTO *K* PARTS (FOLDS), TRAIN ON *K*−1 OF THEM AND TEST THE RESULTING MODEL ON THE LAST ONE. THIS IS CALLED **K-FOLD VALIDATION**. THERE ARE MANY VARIATIONS. HERE ARE JUST TWO:
  - *STRATIFIED K-FOLD VALIDATION*: RESTRICT THE FOLDS TO HAVE THE SAME PERCENTAGE OF CLASSES AS THE FULL SAMPLE
  - *LEAVE-ONE-OUT*: IF *K=N* K-FOLD VALIDAITON

SUMMARY

- FORMULATED A PROBLEM OF PREDICTING NATIVE ADS
- DISCUSSED DATA PREPROCESSING, FEATURE ENGINEERING AND CLASSIFICATION STEPS
- DISCUSSED THE LIBRARIES AVAILABLE TO PERFORM MACHINE LEARNING AT SCALE WITH SPARK
- PROVIDED SOLUTION BASED ON SPARK.ML AND DATAFRAMES: HTTPS://GITHUB.COM/ASVYATKOVSKIY/MLMEETUP2016