A/B testing

Use statistic tools to compare two version of a web page for example headlines, fonts or any web products to see which one performs better. [Process](https://vwo.com/ab-testing/):

* study your web data
* observe customer behavior
* construct a hypothesis
* test your hypothesis
* analyze the data and draw conclusions
* report result

why sometime result is bad: 1. bad setup 2.shock of newness new to predetermine how many clicks

https://www.youtube.com/watch?v=6TI-gQhsf40

accuracy -> compare true positive and true negative.

[ACC](https://en.wikipedia.org/wiki/Confusion_matrix) = (TP+TN)/(P+N)

activation function

in deep learning, the activation function of a node defines the output of that node given an input or set if inputs.  
  
AdaGrad

A sophisticated gradient descent algorithm that rescales the gradients of each parameter, effectively giving each parameter an independent [**learning rate**](https://developers.google.com/machine-learning/glossary/#learning_rate). For a full explanation, see [this paper](http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf).

AUC (Area under the ROC Curve)

ROC curve is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True positive rate, False positive rate.

[AUC](https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc) is an area, AUC provides an aggregate measure of performance across all possible classification thresholds.  
  
automation bias

Errors of automation bias tend to occur when decision-making is dependent on computers or other automated aids and the human is in an observatory role.

back-propagation

BP is a method used in artificial neural network to calculate a gradient that is needed in the weights to be used in the network  
  
baseline

A baseline result is the simplest possible prediction. It is a method that uses heuristics, simple summary statistics, randomness, of learning to create predictions for a dataset. For some problems, this maybe a randam result, and in others it may be the most common prediction.

batch

[Batch](https://www.quora.com/What-is-meant-by-Batch-in-machine-learning) means a group of traning samples, means how training samples are used while computing the loss function. It is distinct form “online” and “mini-batch” learning

batch size

number of training examples utilized/propagated in one iteration.

bias (math)

in probability, “bias” measures the possible outcomes are not equal to real value.

bias (ethics/fairness)

Bias is disproportionate weight in favor of or against one thing, person, or group compared with another. Usually in a way considered to be unfair.

binary classification

classify/group the elements of a given set into to groups/classes.

binning

[Binning](https://docs.tibco.com/pub/spotfire/7.0.1/doc/html/bin/bin_what_is_binning.htm) is a way to group a number of more or less continuous values into a smaller number of “bins”

bucketing

same as bining

calibration layer

A post-prediction adjustment, typically to account for [**prediction bias**](https://developers.google.com/machine-learning/glossary/#prediction_bias). The adjusted predictions and probabilities should match the distribution of an observed set of labels.  
  
candidate sampling

A training-time optimization in which a probability is calculated for all the positive labels, using, for example, softmax, but only for a random sample of negative labels. For example, if we have an example labeled *beagle* and *dog* candidate sampling computes the predicted probabilities and corresponding loss terms for the *beagle* and *dog* class outputs in addition to a random subset of the remaining classes (*cat*, *lollipop*, *fence*). The idea is that the [**negative classes**](https://developers.google.com/machine-learning/glossary/#negative_class) can learn from less frequent negative reinforcement as long as [**positive classes**](https://developers.google.com/machine-learning/glossary/#positive_class) always get proper positive reinforcement, and this is indeed observed empirically. The motivation for candidate sampling is a computational efficiency win from not computing predictions for all negatives.

categorical data

Categorical data represents types of data which may be divided into groups. Basically it means the group belonging to.

centroid

the center of mass of the object, center of a cluster

checkpoint

Data that captures the state of the variables of a model at a particular time. Checkpoints enable exporting model [**weights**](https://developers.google.com/machine-learning/glossary/#weight), as well as performing training across multiple sessions. Checkpoints also enable training to continue past errors (for example, job preemption). Note that the [**graph**](https://developers.google.com/machine-learning/glossary/#graph) itself is not included in a checkpoint.

class

Group  
class-imbalanced data set

The number of observations belonging to one class is significantly different than those belonging to the other class.

classification model

Build a [model](https://www-users.cs.umn.edu/~kumar001/dmbook/ch4.pdf) to assign objects to one of several predefined categories.   
  
classification threshold

Decision threshold, in order to map a logistic regression value to a binary category, toy must have such a classification threshold. A value above that threshold indicates a class, otherwise, it belongs to another class.

clustering

we don't know the classes before we build the model, we need to group similar objects into the same group.

collaborative filtering

[Collaborative filtering](https://towardsdatascience.com/various-implementations-of-collaborative-filtering-100385c6dfe0) is based on assumption that people like things similar to other things they like and things that are liked by other ppl with similar taste.  
  
confirmation bias

It is the tendency to search for, interpret, favor and recall information in a way that confirm one’s preexisting beliefs or hypo.

confusion matrix

It's a table used to measure the performance of model, we have real true and false value and predicted true and false value here.

continuous feature

a variable that has a infinite umber of possible values. For example time, distance, weight  
  
convergence

a state that reached during iteration in which training loss and validation loss change very little or even not at all with each iteration after a certain number of iterations.   
  
convex function

A function in which the region above the graph of the function is a [**convex set**](https://developers.google.com/machine-learning/glossary/#convex_set). The prototypical convex function is shaped something like the letter **U**. For example, the following are all convex functions:

convex optimization

The process of using mathematical techniques such as [**gradient descent**](https://developers.google.com/machine-learning/glossary/#gradient_descent) to find the minimum of a [**convex function**](https://developers.google.com/machine-learning/glossary/#convex_function). A great deal of research in machine learning has focused on formulating various problems as convex optimization problems and in solving those problems more efficiently.

For complete details, see Boyd and Vandenberghe, [Convex Optimization](https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf).

convex set  
  
[convolution](http://forums.fast.ai/t/dense-vs-convolutional-vs-fully-connected-layers/191)

applying a convolution filter to the input matrix which is usually has a much higher dimensionality than that of the convolution filter. Basically it turns a large matrix to a smaller one.

convolutional filter

a square matrix whose values have to be learned by a neural network training algorithm. Convolutional filters are parts of convolutional neural networks where they are used by multiplying their values by the values of the corresponding patches of the input (or the previous layer) and then summing the obtained values.

convolutional layer

A layer of a deep neural network in which a [**convolutional filter**](https://developers.google.com/machine-learning/glossary/#convolutional_filter) passes along an input matrix.

convolutional neural network

A CNN is a class of feedforward neural network, most commonly applied to analyze images. It uses convolutional layer that filter inputs for useful information. convolutional layer have parameters that are learned so that filters are adjusted automatically to extract the most useful information for the task at hand.

Pooling also reduces the memory consumption and thus allows for the usage of more convolutional layers.

convolutional operation

Long story short, use a convolutional layer to scan a image to extract useful information.

cost

difference between target value and predictive value  
  
coverage bias

[Coverage bias](https://en.wikipedia.org/wiki/Survey_sampling) can occur when population members do not appear in the sample frame

cross-entropy

[Cross-entropy](https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html) loss, or log loss, measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.

custom Estimator #TensorFlow

#TensorFlow

An [**Estimator**](https://developers.google.com/machine-learning/glossary/#Estimators) that you write yourself by following [these directions](https://www.tensorflow.org/get_started/custom_estimators).

data analysis  
  
DataFrame

It is a 2-dimensional labeled data structure with columns of potentially different type.

dataset

a collection of examples

decision boundary

In a *classification* problem with two or more classes, a *decision boundary* is a hypersurface that partitions the underlying vector space into two or more regions, one for each *class*.

dense layer

[Dense layer](http://forums.fast.ai/t/dense-vs-convolutional-vs-fully-connected-layers/191/3) is a linear operation in which every input is connected to every output by a weight (so there are n\_inputs \* n\_outputs weights - which can be a lot!). Generally followed by a non-linear activation function

deep model

A type of [**neural network**](https://developers.google.com/machine-learning/glossary/#neural_network) containing multiple [**hidden layers**](https://developers.google.com/machine-learning/glossary/#hidden_layer).

dense feature

A [**feature**](https://developers.google.com/machine-learning/glossary/#feature) in which most values are non-zero, typically a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor) of floating-point values. Contrast with [**sparse feature**](https://developers.google.com/machine-learning/glossary/#sparse_features).

device #TensorFlow

A category of hardware that can run a TensorFlow session, including CPUs, GPUs, and [**TPUs**](https://developers.google.com/machine-learning/glossary/#TPU).

discrete feature

A [**feature**](https://developers.google.com/machine-learning/glossary/#feature) with a finite set of possible values. For example, a feature whose values may only be *animal*, *vegetable*, or *mineral* is a discrete (or categorical) feature. Contrast with [**continuous feature**](https://developers.google.com/machine-learning/glossary/#continuous_feature).

dropout regularization

It prevents *neurons* from co-adapting by randomly setting a fraction of them to 0 at each training iteration.

dynamic model

A [**model**](https://developers.google.com/machine-learning/glossary/#model) that is trained online in a continuously updating fashion. That is, data is continuously entering the model.

early stopping

*Early stopping* is a *regularization* method that involves ending model training before *training loss* finishes decreasing. In early stopping, the engineer ends model training when the *validation loss* starts to increase, that is when generalization performance worsens.

embeddings

An embedding maps an input representation, such as word, sentence or even images, into a vector. Most frequently embeddings refer to word embedding such as **word2vec** or **Glove**

empirical risk minimization (ERM)

A merger of the predictions of multiple [**models**](https://developers.google.com/machine-learning/glossary/#model). You can create an ensemble via one or more of the following:

* different initializations
* different [**hyperparameters**](https://developers.google.com/machine-learning/glossary/#hyperparameter)
* different overall structure

[Deep and wide models](https://www.tensorflow.org/tutorials/wide_and_deep) are a kind of ensemble.

ensemble

learning a strong classifier by combining multiple weak classifier  
  
epoch

One passthrough the training set by a machine learning algorithm  
  
Estimator #tensorflow

An instance of the tf.Estimator class, which encapsulates logic that builds a TensorFlow graph and runs a TensorFlow session. You may create your own [**custom Estimators**](https://developers.google.com/machine-learning/glossary/#custom_estimator) (as described [here](https://www.tensorflow.org/extend/estimators)) or instantiate [**premade Estimators**](https://developers.google.com/machine-learning/glossary/#premade_Estimator)created by others.

example #features

Also called instance is a member of a dataset, typically an example is a vector of features, each features represents some specific property of the example.   
  
experimenter's bias #fairness  
  
false negative (FN)

Model mistakenly predict the negative class, for example, the model inferred that a particular email message was not spam, but the email message actually was spam.

false positive (FP)

See above. mistakenly predict the positive clas.

feature

an attribute of an example, usually a part of a feature vector. if an example is a people, it can have the following features: height, weight, race, etc.

feature cross

A [**synthetic feature**](https://developers.google.com/machine-learning/glossary/#synthetic_feature) formed by crossing (taking a [**Cartesian product**](https://wikipedia.org/wiki/Cartesian_product) of) individual binary features obtained from[**categorical data**](https://developers.google.com/machine-learning/glossary/#categorical_data) or from [**continuous features**](https://developers.google.com/machine-learning/glossary/#continuous_feature) via [**bucketing**](https://developers.google.com/machine-learning/glossary/#bucketing). Feature crosses help represent nonlinear relationships.

feature engineering

The process of determining which [**features**](https://developers.google.com/machine-learning/glossary/#feature) might be useful in training a model, and then converting raw data from log files and other sources into said features. In TensorFlow, feature engineering often means converting raw log file entries to **[tf.Example](https://developers.google.com/machine-learning/glossary/" \l "tf.Example)** protocol buffers. See also [tf.Transform](https://github.com/tensorflow/transform).

Feature engineering is sometimes called **feature extraction**.

feature set

The group of [**features**](https://developers.google.com/machine-learning/glossary/#feature) your machine learning model trains on. For example, postal code, property size, and property condition might comprise a simple feature set for a model that predicts housing prices.

feature selection

A progress of removing the dataset features that seem irrelevant for modeling.

feature spec #TensorFlow

Describes the information required to extract [**features**](https://developers.google.com/machine-learning/glossary/#feature) data from the **[tf.Example](https://developers.google.com/machine-learning/glossary/" \l "tf.Example)** protocol buffer. Because the tf.Example protocol buffer is just a container for data, you must specify the following:

* the data to extract (that is, the keys for the features)
* the data type (for example, float or int)
* The length (fixed or variable)

The [**Estimator API**](https://developers.google.com/machine-learning/glossary/#Estimators) provides facilities for producing a feature spec from a list of **[FeatureColumns](https://developers.google.com/machine-learning/glossary/" \l "feature_columns)**.

feedforward neural network

a neural network wherein connections between the units do not form a cycle. As such, it is different from recurrent neural networks. The information in this network moves in only one direction, forward from the input nodes, through the hidden nodes and to the output nodes.  
  
few-shot learning

usually employed in *classification*, designed to learn effective classifiers from only a *small numbe*r of training examples.  
  
full softmax -> compare with candidate sampling.

See **[softmax](https://developers.google.com/machine-learning/glossary/" \l "softmax)**. Contrast with [**candidate sampling**](https://developers.google.com/machine-learning/glossary/#candidate_sampling).

fully connected layer

A [**hidden layer**](https://developers.google.com/machine-learning/glossary/#hidden_layer) in which each [**node**](https://developers.google.com/machine-learning/glossary/#node) is connected to *every* node in the subsequent hidden layer.

A fully connected layer is also known as a [**dense layer**](https://developers.google.com/machine-learning/glossary/#dense_layer).

gated recurrent unit (GRU)

a simplified version of an LSTM unit with **fewer parameters**. Just like an LSTM cell, it uses a gating mechanism to allow RNNs to efficiently learn long-range dependencies by reducing the vanishing gradient problem. The GRU consist of a **reset and update gate** that determine which the extent to which the unit should keep the old value and to which the new value has to replace it at the current time step.

generalization

Refers to your model's ability to make correct predictions on new, previously unseen data as opposed to the data used to train the model.

generalized linear model

A generalization of [**least squares regression**](https://developers.google.com/machine-learning/glossary/#least_squares_regression) models, which are based on [Gaussian noise](https://wikipedia.org/wiki/Gaussian_noise), to other types of models based on other types of noise, such as [Poisson noise](https://wikipedia.org/wiki/Shot_noise) or categorical noise.

gradient

The vector of [**partial derivatives**](https://developers.google.com/machine-learning/glossary/#partial_derivative) with respect to all of the independent variables. In machine learning, the gradient is the vector of partial derivatives of the model function. The gradient points in the direction of steepest ascent.

gradient clipping

**capping gradient value** before applying them in gradient descent algorithm during backpropagation. Gradient clipping helps ensure numerical stability an **prevents exploding gradient problem**.  
  
gradient descent

**an iterative optimization algorithm for differentiable functions**. It is designed to find the **minimum** of a function. In many machine learning algorithms, gradient descent, or its variant is used to find the **minimum of the** **loss function** given the training dataset.

graph #TensorFlow

In TensorFlow, a computation specification. Nodes in the graph represent operations. Edges are directed and represent passing the result of an operation (a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor)) as an operand to another operation. Use **[TensorBoard](https://developers.google.com/machine-learning/glossary/" \l "TensorBoard)** to visualize a graph.

grid search

*Grid search* is a way of **hyperparameter tuning**. The process consists of training the *model* on all possible combinations of hyperparameter values and then selecting the best combination. The best combination of hyperparameters is the one that performs the best on the *validation set*.

group attribution bias -> comapre homogeneity bias and in-group bias.

Assuming that what is true for an individual is also true for everyone in that group. The effects of group attribution bias can be exacerbated if a [convenience sample](https://wikipedia.org/wiki/Convenience_sampling) is used for data collection. In a non-representative sample, attributions may be made that do not reflect reality.

See also [**out-group homogeneity bias**](https://developers.google.com/machine-learning/glossary/#out-group_homogeneity_bias) and [**in-group bias**](https://developers.google.com/machine-learning/glossary/#in-group_bias).

heuristic

A practical and nonoptimal solution to a problem, which is sufficient for making progress or for learning from.

hidden layer

layers between input layers and output layers, where neurons take in a set of weighted inputs and produce an output through **an activation function**.

Hierarchical Clustering Algorithm

[*Hierarchical clustering algorithms*](https://semanti.ca/blog/?glossary-of-machine-learning-terms)is a category of *clustering algorithms* that create a tree of clusters. Hierarchical clustering algorithms are well-suited to hierarchical data, such as botanical taxonomies.

hinge loss

a loss function used for training classifiers. The hinge loss is used for “maximum-margin” classification, most notably for support vector machine.

Loss(y) = max(0, 1-y\*y^), y is the “raw” output of the classifier’s decision function, not the predicted class label.

holdout data

A part of the dataset that contains examples intentionally **not used during training**. **Validation set and test** set are examples of holdout data

hyperparameter

A parameter of a machine learning algorithm **whose value is not optimized during training**. It can be training iterations, the size of minibatch, a regularization parameter, the value of the learning rate, and many others.

Hyperparameter **can be optimized** using cross-validation and techniques like grid search, random search, Bayesian optimization, evolutionary optimization, and others.   
  
hyperplane

A boundary that **separates a space into two subspaces**. For example, a line is a hyperplane in two dimensions and a plane is a hyperplane in three dimensions. In machine learning, a hyperplane is usually a boundary separating a high-dimensional space.

implicit bias #fairness

Automatically making an association or assumption based on one’s mental models and memories. Implicit bias can affect the following:

* How data is collected and classified.
* How ML systems are designed and developed.

independently and identically distributed (i.i.d)

Data drawn from a distribution that doesn't change, and where each value drawn doesn't depend on values that have been drawn previously. An i.i.d. is the [ideal gas](https://wikipedia.org/wiki/Ideal_gas) of machine learning—a useful mathematical construct but almost never exactly found in the real world. For example, the distribution of visitors to a web page may be i.i.d. over a brief window of time; that is, the distribution doesn't change during that brief window and one person's visit is generally independent of another's visit. However, if you expand that window of time, seasonal differences in the web page's visitors may appear.

inference

In machine learning, often refers to the process of making predictions by applying the trained model to [**unlabeled examples**](https://developers.google.com/machine-learning/glossary/#unlabeled_example). In statistics, inference refers to the process of fitting the parameters of a distribution conditioned on some observed data. (See the [Wikipedia article on statistical inference](https://wikipedia.org/wiki/Statistical_inference).)

in-group bias

Showing partiality to one's own group or own characteristics. If testers or raters consist of the machine learning developer's friends, family, or colleagues, then in-group bias may invalidate product testing or the data set.

In-group bias is a form of [**group attribution bias**](https://developers.google.com/machine-learning/glossary/#group_attribution_bias). See also [**out-group homogeneity bias**](https://developers.google.com/machine-learning/glossary/#out-group_homogeneity_bias).

input function #TensorFlow

In TensorFlow, a function that returns input data to the training, evaluation, or prediction method of an [**Estimator**](https://developers.google.com/machine-learning/glossary/#Estimators). For example, the training input function returns a [**batch**](https://developers.google.com/machine-learning/glossary/#batch) of features and labels from the [**training set**](https://developers.google.com/machine-learning/glossary/#training_set).

input layer

a layer whose neurons take as **input the features of the input instance**.

instance

See examples  
  
interpretability

The degree to which a model's predictions can be readily explained. Deep models are often non-interpretable; that is, a deep model's different layers can be hard to decipher. By contrast, linear regression models and [**wide models**](https://developers.google.com/machine-learning/glossary/#wide_model) are typically far more interpretable.

inter-rater agreement

A measurement of how often human raters agree when doing a task. If raters disagree, the task instructions may need to be improved. Also sometimes called **inter-annotator agreement** or **inter-rater reliability**. See also [Cohen's kappa](https://wikipedia.org/wiki/Cohen%27s_kappa), which is one of the most popular inter-rater agreement measurements.

iteration

**Number of times** the machine learning algorithm’s parameters are updated while training a model on a dataset.

k-means

clustering data into exactly k clusters.

First, define k initial cluster *centroids*.

Second, assign each example to the closest centroid.

Third, recompute the new position for each centroid as the average of the examples assigned to it.

Iterate back to step two.

K Nearest Neighbors

An *instance-based learning algorithm* that can be used for both classification and regression

When used in the *classification* context, the algorithm predicts the class of an *unlabeled example* as the majority class among kk its closest neighbors in the vector space. In the *regression* context, the label of an unlabeled example is calculated as an average of the labels of the kk its closest neighbors. The distance from one example to another is usually given by a *similarity metric*.

k-median

A clustering algorithm closely related to [**k-means**](https://developers.google.com/machine-learning/glossary/#k-means). The practical difference between the two is as follows:

* In k-means, centroids are determined by minimizing the sum of the *squares* of the distance between a centroid candidate and each of its examples.
* In k-median, centroids are determined by minimizing the sum of the distance between a centroid candidate and each of its examples.

Keras

A popular Python machine learning API. [Keras](https://keras.io/) runs on several deep learning frameworks, including TensorFlow, where it is made available as **[tf.keras](https://www.tensorflow.org/api_docs/python/tf/keras)**.

Kernel Support Vector Machines (KSVMs)

the function of kernel is to take data as input and transform it into the required form.

We need to create hyperspace to separate a space into subspace

For example,*linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid.*

L1 loss

[**Loss**](https://developers.google.com/machine-learning/glossary/#loss) function based on the absolute value of the difference between the values that a model is predicting and the actual values of the [**labels**](https://developers.google.com/machine-learning/glossary/#label). L1 loss is less sensitive to outliers than [**L2 loss**](https://developers.google.com/machine-learning/glossary/#squared_loss).

L1 regularization

A type of [**regularization**](https://developers.google.com/machine-learning/glossary/#regularization) that penalizes weights in proportion to the sum of the absolute values of the weights. In models relying on [**sparse features**](https://developers.google.com/machine-learning/glossary/#sparse_features), L1 regularization helps drive the weights of irrelevant or barely relevant features to exactly 0, which removes those features from the model. Contrast with [**L2 regularization**](https://developers.google.com/machine-learning/glossary/#L2_regularization).  
  
L2 loss

Squared loss

L2 regularization

A type of [**regularization**](https://developers.google.com/machine-learning/glossary/#regularization) that penalizes weights in proportion to the sum of the *squares* of the weights. L2 regularization helps drive outlier weights (those with high positive or low negative values) closer to 0 but not quite to 0. (Contrast with [**L1 regularization**](https://developers.google.com/machine-learning/glossary/#L1_regularization).) L2 regularization always improves generalization in linear models.

label

In supervised learning, the "answer" or "result" portion of an [**example**](https://developers.google.com/machine-learning/glossary/#example). Each example in a labeled data set consists of one or more features and a label. For instance, in a housing data set, the features might include the number of bedrooms, the number of bathrooms, and the age of the house, while the label might be the house's price. In a spam detection dataset, the features might include the subject line, the sender, and the email message itself, while the label would probably be either "spam" or "not spam."

labeled example

An example that contains [**features**](https://developers.google.com/machine-learning/glossary/#feature) and a [**label**](https://developers.google.com/machine-learning/glossary/#label). In supervised training, models learn from labeled examples.

lambda

Regularization term

layer

A set of [**neurons**](https://developers.google.com/machine-learning/glossary/#neuron) in a [**neural network**](https://developers.google.com/machine-learning/glossary/#neural_network) that process a set of input features, or the output of those neurons.

Also, an abstraction in TensorFlow. Layers are Python functions that take [**Tensors**](https://developers.google.com/machine-learning/glossary/#tensor) and configuration options as input and produce other tensors as output. Once the necessary Tensors have been composed, the user can convert the result into an [**Estimator**](https://developers.google.com/machine-learning/glossary/#Estimators) via a [**model function**](https://developers.google.com/machine-learning/glossary/#model_function).  
  
Layers API #TensorFlow

A TensorFlow API for constructing a [**deep**](https://developers.google.com/machine-learning/glossary/#deep_model) neural network as a composition of layers. The Layers API enables you to build different types of [**layers**](https://developers.google.com/machine-learning/glossary/#layer), such as:

* tf.layers.Dense for a [**fully-connected layer**](https://developers.google.com/machine-learning/glossary/#fully_connected_layer).
* tf.layers.Conv2D for a convolutional layer.

When writing a [**custom Estimator**](https://developers.google.com/machine-learning/glossary/#custom_estimator), you compose Layers objects to define the characteristics of all the [**hidden layers**](https://developers.google.com/machine-learning/glossary/#hidden_layers).

The Layers API follows the [**Keras**](https://developers.google.com/machine-learning/glossary/#Keras) layers API conventions. That is, aside from a different prefix, all functions in the Layers API have the same names and signatures as their counterparts in the Keras layers API.

learning rate

A **scalar** used to update model parameters via gradient descent.

Gradient step = gradient descent algorithm \* gradient by the learning rate

least squares regression

A linear regression model trained by minimizing [**L2 Loss**](https://developers.google.com/machine-learning/glossary/#L2_loss).

linear regression

A type of [**regression model**](https://developers.google.com/machine-learning/glossary/#regression_model) that outputs a continuous value from a linear combination of input features.

logistic regression

A model that generates a probability for each possible discrete label value in classification problems by applying a [**sigmoid function**](https://developers.google.com/machine-learning/glossary/#sigmoid_function) to a linear prediction. Although logistic regression is often used in [**binary classification**](https://developers.google.com/machine-learning/glossary/#binary_classification) problems, it can also be used in [**multi-class**](https://developers.google.com/machine-learning/glossary/#multi-class) classification problems (where it becomes called **multi-class logistic regression** or **multinomial regression**).

logits

The vector of raw (non-normalized) predictions that a classification model generates, which is ordinarily then passed to a normalization function. If the model is solving a multi-class classification problem, logits typically become an input to the[softmax function](https://www.tensorflow.org/api_docs/python/tf/nn/softmax_cross_entropy_with_logits_v2). The softmax function then generates a vector of (normalized) probabilities with one value for each possible class.

In addition, logits sometimes refer to the element-wise inverse of the [**sigmoid function**](https://developers.google.com/machine-learning/glossary/#sigmoid_function). For more information, see[tf.nn.sigmoid\_cross\_entropy\_with\_logits](https://www.tensorflow.org/api_docs/python/tf/nn/sigmoid_cross_entropy_with_logits).

Log Loss

loss function used in the binary logistic regression

logloss(y) = -(ylog(p)+(1-y)log(1-p))

p is the probability predicted by the model, y is true label

Long short-term memory unit (LSTM)

Long short-term memory (LSTM) units in recurrent neural networks help reducing the vanishing gradient problem during the backpropagation. LSTM unit is a neuron that has a memory cell and three gates: "input", "output" and "forget". The purpose of the memory cell is to retain information previously used by the RNN or forget if needed. Neural networks with LSTM units, also called LSTM networks, are explicitly designed to avoid the long-term dependency problem in RNNs and have been shown to be able to learn complex sequences better than simple RNNs.

The structure of a memory cell is: an input gate, that determines how much of information from the previous layer gets stored in the cell; the output gate, that determines how of the next layer gets to know about the state of the current cell; and the forget gate, which determines what to forget about the current state of the memory cell.

log-odds  
  
loss function

In the *classification* context: describes the cost of assigning the *label* y^ to a sample of *class* y.

In the *regression* context: describes the cost of assigning the value y^ to the regression function evaluated at →xx→, where it takes value y.

machine learning

algorithms that learn from, make decisions and predictions based on data.

Mean Squared Error (MSE)

The average squared loss per example. MSE is calculated by dividing the [**squared loss**](https://developers.google.com/machine-learning/glossary/#squared_loss) by the number of [**examples**](https://developers.google.com/machine-learning/glossary/#example). The values that [**TensorFlow Playground**](https://developers.google.com/machine-learning/glossary/#TensorFlow_Playground) displays for "Training loss" and "Test loss" are MSE.  
  
metric

[Metrics](https://towardsdatascience.com/metrics-to-evaluate-your-machine-learning-algorithm-f10ba6e38234) is for measuring the performance of our model.

Classification Accuracy

Logarithmic Loss

Confusion Matrix

Area under Curve

F1 Score

Mean Absolute Error

Mean Squared Error

mini-batch

see batch

mini-batch stochastic gradient descent (SGD)

A [**gradient descent**](https://developers.google.com/machine-learning/glossary/#gradient_descent) algorithm that uses [**mini-batches**](https://developers.google.com/machine-learning/glossary/#mini-batch). In other words, mini-batch SGD estimates the gradient based on a small subset of the training data. [**Vanilla SGD**](https://developers.google.com/machine-learning/glossary/#SGD) uses a mini-batch of size 1.

ML  
  
model

Statistical model is the result of a machine learning algorithm applied to the training data.

Model is often a parametrized mathematical formula, where parameters are learning by the machine learning algorithm.

model function #TensorFlow

The function within an [**Estimator**](https://developers.google.com/machine-learning/glossary/#Estimators) that implements ML training, evaluation, and inference. For example, the training portion of a model function might handle tasks such as defining the topology of a deep neural network and identifying its [**optimizer**](https://developers.google.com/machine-learning/glossary/#optimizer) function. When using [**premade Estimators**](https://developers.google.com/machine-learning/glossary/#premade_Estimator), someone has already written the model function for you. When using [**custom Estimators**](https://developers.google.com/machine-learning/glossary/#custom_estimator), you must write the model function yourself.

model training

The process of determining the best [**model**](https://developers.google.com/machine-learning/glossary/#model).

multi-class classification

*Multi-class classification* is a *classification* problem that distinguishes among more than two classes.

multinomial classification

Multinomial regression*is a variant of the*logistic regression*algorithm used for*multi-class classification*problems.*

negative class

In [**binary classification**](https://developers.google.com/machine-learning/glossary/#binary_classification), one class is termed positive and the other is termed negative. The positive class is the thing we're looking for and the negative class is the other possibility. For example, the negative class in a medical test might be "not tumor." The negative class in an email classifier might be "not spam." See also [**positive class**](https://developers.google.com/machine-learning/glossary/#positive_class).

neural network

A model that, taking inspiration from the brain, is composed of layers (at least one of which is [**hidden**](https://developers.google.com/machine-learning/glossary/#hidden_layer)) consisting of simple connected units or [**neurons**](https://developers.google.com/machine-learning/glossary/#neuron) followed by nonlinearities.

neuron

also called unit is a node in a neural network, typically taking in multiple input values and generate one output value.

The neuron calculates the output value by applying an *activation function* (nonlinear transformation) to a weighted sum of input values.

node (neural network)  
  
node (TensorFlow graph)  
  
non-response bias

See [**selection bias**](https://developers.google.com/machine-learning/glossary/#selection_bias).

normalization

the process of converting an actual range of values into a standard range of values, e.g [-1, 1], [0,1]

numerical data

[**Features**](https://developers.google.com/machine-learning/glossary/#feature) represented as integers or real-valued numbers. For example, in a real estate model, you would probably represent the size of a house (in square feet or square meters) as numerical data. Representing a feature as numerical data indicates that the feature's values have a *mathematical* relationship to each other and possibly to the label. For example, representing the size of a house as numerical data indicates that a 200 square-meter house is twice as large as a 100 square-meter house. Furthermore, the number of square meters in a house probably has some mathematical relationship to the price of the house.

numpy  
  
objective

A metric that your algorithm is trying to optimize.

one-hot encoding

transforming a **categorical feature** into a vector of several binary features where all components are 00, except for one component with a value of 1.

 if our *example* has a categorical feature "weather" and this feature has three possible values: "sun", "rain", "clouds", then to transform this feature into something a *machine learning algorithm* that can only work with numerical values, one can transform this feature into a vector of three numerical values:

sun=[1,0,0]

rain=[0,1,0]

clouds=[0,0,1]

one-shot learning

training a *model* with only a single *example* per *class*.

*One way to build a system capable of one-shot learning is to use*representation learning*, to learn representations or*features*of data that can be used to accurately classify a single example.*

one-vs.-all

Given a classification problem with N possible solutions, a one-vs.-all solution consists of N separate [**binary classifiers**](https://developers.google.com/machine-learning/glossary/#binary_classification)—one binary classifier for each possible outcome. For example, given a model that classifies examples as animal, vegetable, or mineral, a one-vs.-all solution would provide the following three separate binary classifiers:

* animal vs. not animal
* vegetable vs. not vegetable
* mineral vs. not mineral

Operation (op) #TensorFlow

A node in the TensorFlow graph. In TensorFlow, any procedure that creates, manipulates, or destroys a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor) is an operation. For example, a matrix multiply is an operation that takes two Tensors as input and generates one Tensor as output.

optimizer  
A specific implementation of the [**gradient descent**](https://developers.google.com/machine-learning/glossary/#gradient_descent) algorithm. TensorFlow's base class for optimizers is [tf.train.Optimizer](https://www.tensorflow.org/api_docs/python/tf/train/Optimizer). Different optimizers may leverage one or more of the following concepts to enhance the effectiveness of gradient descent on a given [**training set**](https://developers.google.com/machine-learning/glossary/#training_set):

* [momentum](https://www.tensorflow.org/api_docs/python/tf/train/MomentumOptimizer) (Momentum)
* update frequency ([AdaGrad](https://www.tensorflow.org/api_docs/python/tf/train/AdagradOptimizer) = ADAptive GRADient descent; [Adam](https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer) = ADAptive with Momentum; RMSProp)
* sparsity/regularization ([Ftrl](https://www.tensorflow.org/api_docs/python/tf/train/FtrlOptimizer))
* more complex math ([Proximal](https://www.tensorflow.org/api_docs/python/tf/train/ProximalGradientDescentOptimizer), and others)

You might even imagine an [NN-driven optimizer](https://arxiv.org/abs/1606.04474).

momentum (Momentum)

out-group homogeneity bias -> compare in-group bias.  
  
The tendency to see out-group members as more alike than in-group members when comparing attitudes, values, personality traits, and other characteristics. **In-group** refers to people you interact with regularly; **out-group** refers to people you do not interact with regularly. If you create a data set by asking people to provide attributes about out-groups, those attributes may be less nuanced and more stereotyped than attributes that participants list for people in their in-group.

For example, Lilliputians might describe the houses of other Lilliputians in great detail, citing small differences in architectural styles, windows, doors, and sizes. However, the same Lilliputians might simply declare that Brobdingnagians all live in identical houses.

Out-group homogeneity bias is a form of [**group attribution bias**](https://developers.google.com/machine-learning/glossary/#group_attribution_bias).

See also [**in-group bias**](https://developers.google.com/machine-learning/glossary/#in-group_bias).

outliers

an example that appears far away and diverges from an overall pattern in the dataset.

output layer

The "final" layer of a neural network. The layer containing the answer(s).

overfitting

perform very well in training data, but poor in testing data

*Overfitting* occurs when the machine learning algorithm learns a model that fits the training data too well by incorporating details and noise specific to the training data.

The problem of overfitting is usually solved by ***regularization*** or ***early stopping***,

pandas

A column-oriented data analysis API. Many ML frameworks, including TensorFlow, support pandas data structures as input. See [pandas documentation](http://pandas.pydata.org/).

parameter

A *parameter* of a *model* is the quantity a machine learning algorithm modifies in order to minimize the *loss function*. For example, in *neural networks*, parameters are weights applied to inputs of *neurons*.

partial derivative

A derivative in which all but one of the variables is considered a constant. For example, the partial derivative of *f(x, y)*with respect to *x* is the derivative of *f* considered as a function of *x* alone (that is, keeping *y* constant). The partial derivative of *f* with respect to *x* focuses only on how *x* is changing and ignores all other variables in the equation.

participation bias

non-response bias

performance  
  
perplexity

One measure of how well a [**model**](https://developers.google.com/machine-learning/glossary/#model) is accomplishing its task. For example, suppose your task is to read the first few letters of a word a user is typing on a smartphone keyboard, and to offer a list of possible completion words. Perplexity, P, for this task is approximately the number of guesses you need to offer in order for your list to contain the actual word the user is trying to type.

pipeline

The infrastructure surrounding a machine learning algorithm. A pipeline includes gathering the data, putting the data into training data files, training one or more models, and exporting the models to production.

pooling

reducing **a matrix created by an earlier convolution** to a smaller matrix. Pooling usually involves taking either the maximum or average value across the pooled area.

positive class -> Contrast with negative class.  
  
precision

TP/(TP+FP), how many of these positive predictions were correct.

**How sure** you are its actually a positive or say fraud

recall

TP/(TP+FN), measure how many of the positively labeled examples were correctly classified by the model.

**How many positive / fraud case**s you can catch

prediction

A metric for [**classification models**](https://developers.google.com/machine-learning/glossary/#classification_model). Precision identifies the frequency with which a model was correct when predicting the [**positive class**](https://developers.google.com/machine-learning/glossary/#positive_class).

prediction bias

A model's output when provided with an input [**example**](https://developers.google.com/machine-learning/glossary/#example).

premade Estimator #TensorFlow

An [**Estimator**](https://developers.google.com/machine-learning/glossary/#Estimator) that someone has already built. TensorFlow provides several premade Estimators, including DNNClassifier, DNNRegressor, and LinearClassifier. To learn more about premade Estimators, see [Premade Estimators](https://www.tensorflow.org/get_started/premade_estimators) or [Premade Estimators for ML Beginners](https://www.tensorflow.org/get_started/get_started_for_beginners).

pre-trained model

Models or model components (such as [**embeddings**](https://developers.google.com/machine-learning/glossary/#embeddings)) that have been already been trained. Sometimes, you'll feed pre-trained embeddings into a [**neural network**](https://developers.google.com/machine-learning/glossary/#neural_network). Other times, your model will train the embeddings itself rather than rely on the pre-trained embeddings.

prior belief

What you believe about the data before you begin training on it. For example, [**L2 regularization**](https://developers.google.com/machine-learning/glossary/#L2_regularization) relies on a prior belief that [**weights**](https://developers.google.com/machine-learning/glossary/#weight) should be small and normally distributed around zero.

proxy labels

Data used to approximate labels not directly available in a data set.

For example, suppose you want *is it raining?* to be a Boolean label for your data set, but the data set doesn't contain rain data. If photographs are available, you might establish pictures of people carrying umbrellas as a proxy label for *is it raining?* However, proxy labels may distort results. For example, in some places, it may be more common to carry umbrellas to protect against sun than the rain.

principle component analysis (PCA)

[PCA](https://semanti.ca/blog/?glossary-of-machine-learning-terms) is a linear transformation which projects n examples each consisting of m features on a hyperplane in such a way the projection error is minimal.

Basically, it is trying to find the most important part of a dataset to replace the whole giant dataset.

By keeping only several biggest principal components and the projections of the examples on them, and by discarding the rest of information, one can reconstruct the dataset from a much smaller amount of information (with some small loss in accuracy of reconstruction because of the discarded information).

queue #TensorFlow

A TensorFlow [**Operation**](https://developers.google.com/machine-learning/glossary/#Operation) that implements a queue data structure. Typically used in I/O.

rank (Tensor)

The number of dimensions in a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor). For instance, a scalar has rank 0, a vector has rank 1, and a matrix has rank 2.

rank (ordinality)

The ordinal position of a class in an ML problem that categorizes classes from highest to lowest. For example, a behavior ranking system could rank a dog's rewards from highest (a steak) to lowest (wilted kale).

rater  
  
A human who provides [**labels**](https://developers.google.com/machine-learning/glossary/#label) in [**examples**](https://developers.google.com/machine-learning/glossary/#example). Sometimes called an "annotator."

Recurrent neural network RNN

Fox example, stocks price for today, even though is affected by today’s new info, is also depends a lot of yesterday, two years ago, or even last week.+

sequence generation, text predict depends a lot on previous word and word before.

A neural network that usually deal with sequential data like texts, audio and video.

It contains an internal memory. An RNN has an internal loop that allows information to persist in the network. *Neurons* receive information not just from the previous layer, but also from themselves from the previous pass. This means that the order of inputs to the RNN matter as RNNs have a state.

RNNs are particularly sensitive to the *vanishing* and *exploding gradient problems*, where depending on the *activation functions* used, the information can get lost over time. *Long short-term memory units* (LSTM) addresses this problem. RNNs are commonly used with sequential data, like in natural language processing.

Overcome exploding gradients: truncated BTT, Clip gradients at threshold. RMSprop to adjust learning rate

overcome Vanishing(start affect little on the end): ReLU activation function, RMSprop, LSTM, GRUs

Common examples of **sequential data** includes texts, audio and video.

regression model

a model that outputs continuous data.

regularization  
  
regularization rate

a technique to make the fitted function smoother. This helps to prevent overfitting. The most widely used regularization techniques are **L1**, **L2**, **dropout**, and **weight decay**.

reporting bias

The fact that the frequency with which people write about actions, outcomes, or properties is not a reflection of their real-world frequencies or the degree to which a property is characteristic of a class of individuals. Reporting bias can influence the composition of data that ML systems learn from.

For example, in books, the word *laughed* is more prevalent than *breathed*. An ML model that estimates the relative frequency of laughing and breathing from a book corpus would probably determine that laughing is more common than breathing.

representation - features.

The process of mapping data to useful [**features**](https://developers.google.com/machine-learning/glossary/#feature).

root directory

The directory you specify for hosting subdirectories of the TensorFlow checkpoint and events files of multiple models.

rotational invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the orientation of the image changes. For example, the algorithm can still identify a tennis racket whether it is pointing up, sideways, or down. Note that rotational invariance is not always desirable; for example, an upside-down 9 should not be classified as a 9.

sampling bias  
  
scaling

A commonly used practice in [**feature engineering**](https://developers.google.com/machine-learning/glossary/#feature_engineering) to tame a feature's range of values to match the range of other features in the data set. For example, suppose that you want all floating-point features in the data set to have a range of 0 to 1. Given a particular feature's range of 0 to 500, you could scale that feature by dividing each value by 500.

selection bias  
  
semi-supervised learning

learning a *model* by using both *labeled* and *unlabeled examples*.

Semi-supervised learning techniques take advantage of a small amount of labeled data and a large amount of unlabeled data to produce a better model than a purely supervised learning or a purely unsupervised learning technique.  
  
sequence model

A model whose inputs have a sequential dependence. For example, predicting the next video watched from a sequence of previously watched videos.  
  
sigmoid function  
  
size invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the size of the image changes. For example, the algorithm can still identify a cat whether it consumes 2M pixels or 200K pixels. Note that even the best image classification algorithms still have practical limits on size invariance. For example, an algorithm (or human) is unlikely to correctly classify a cat image consuming only 20 pixels.

softmax

A function that provides probabilities for each possible class in a [**multi-class classification model**](https://developers.google.com/machine-learning/glossary/#multi-class). The probabilities add up to exactly 1.0. For example, softmax might determine that the probability of a particular image being a dog at 0.9, a cat at 0.08, and a horse at 0.02. (Also called **full softmax**.)

sparse feature

[**Feature**](https://developers.google.com/machine-learning/glossary/#feature) vector whose values are predominately zero or empty. For example, a vector containing a single 1 value and a million 0 values is sparse. As another example, words in a search query could also be a sparse feature—there are many possible words in a given language, but only a few of them occur in a given query.

sparse representation

A [**representation**](https://developers.google.com/machine-learning/glossary/#representation) of a tensor that only stores nonzero elements.

For example, the English language consists of about a million words. Consider two ways to represent a count of the words used in one English sentence:

* A **dense representation** of this sentence must set an integer for all one million cells, placing a 0 in most of them, and a low integer into a few of them.
* A sparse representation of this sentence stores only those cells symbolizing a word actually in the sentence. So, if the sentence contained only 20 unique words, then the sparse representation for the sentence would store an integer in only 20 cells.

sparsity  
  
spatial pooling  
  
squared hinge loss

The square of the [**hinge loss**](https://developers.google.com/machine-learning/glossary/#hinge-loss). Squared hinge loss penalizes outliers more harshly than regular hinge loss.  
  
squared loss

The [**loss**](https://developers.google.com/machine-learning/glossary/#loss) function used in [**linear regression**](https://developers.google.com/machine-learning/glossary/#linear_regression). (Also known as **L2 Loss**.) This function calculates the squares of the difference between a model's predicted value for a labeled [**example**](https://developers.google.com/machine-learning/glossary/#example) and the actual value of the [**label**](https://developers.google.com/machine-learning/glossary/#label). Due to squaring, this loss function amplifies the influence of bad predictions. That is, squared loss reacts more strongly to outliers than [**L1loss**](https://developers.google.com/machine-learning/glossary/#L1_loss).

static model

A model that is trained offline.

stationarity

A property of data in a data set, in which the data distribution stays constant across one or more dimensions. Most commonly, that dimension is time, meaning that data exhibiting stationarity doesn't change over time. For example, data that exhibits stationarity doesn't change from September to December.

step  
  
step size  
  
stochastic gradient descent (SGD)

a type of *gradient descent* algorithm where the gradient of the function to be optimized is computed by taking a sample of data. The update to the coefficients is performed for each training instance, rather than at the end of the batch of instances.

The learning can be much faster with stochastic gradient descent for very large training datasets and often one only need a small number of passes through the dataset (one pass through the dataset is called *epoch*) to reach a good or good enough set of coefficients.

structural risk minimization (SRM)

An algorithm that balances two goals:

* The desire to build the most predictive model (for example, lowest loss).
* The desire to keep the model as simple as possible (for example, strong regularization).

For example, a function that minimizes loss+regularization on the training set is a structural risk minimization algorithm

Support Vector Machine (SVM)

 a classification algorithm that seeks to maximize the margin between positive and negative classes. SVM is often used together with kernels, functions that map input examples (given as multidimensional vectors) to a higher dimensional space.

Use **hinge loss** as loss function

SVD

Its used to find which feature is the most important

stride

In a convolutional operation or pooling, the delta in each dimension of the next series of input slices. For example, the following animation demonstrates a (1,1) stride during a convolutional operation. Therefore, the next input slice starts one position to the right of the previous input slice. When the operation reaches the right edge, the next slice is all the way over to the left but one position down.

subsampling  
  
summary in TensorFlow  
  
supervised machine learning

Training a [**model**](https://developers.google.com/machine-learning/glossary/#model) from input data and its corresponding [**labels**](https://developers.google.com/machine-learning/glossary/#label). Supervised machine learning is analogous to a student learning a subject by studying a set of questions and their corresponding answers. After mastering the mapping between questions and answers, the student can then provide answers to new (never-before-seen) questions on the same topic. Compare with [**unsupervised machine learning**](https://developers.google.com/machine-learning/glossary/#unsupervised_machine_learning).

synthetic feature

A [**feature**](https://developers.google.com/machine-learning/glossary/#feature) not present among the input features, but created from one or more of them. Kinds of synthetic features include:

* [**Bucketing**](https://developers.google.com/machine-learning/glossary/#bucketing) a continuous feature into range bins.
* Multiplying (or dividing) one feature value by other feature value(s) or by itself.
* Creating a [**feature cross**](https://developers.google.com/machine-learning/glossary/#feature_cross).

Features created by [**normalizing**](https://developers.google.com/machine-learning/glossary/#normalization) or [**scaling**](https://developers.google.com/machine-learning/glossary/#scaling) alone are not considered synthetic features.

target  
  
temporal data

Data recorded at different points in time. For example, winter coat sales recorded for each day of the year would be temporal data.  
  
Tensor

The primary data structure in TensorFlow programs. Tensors are N-dimensional (where N could be very large) data structures, most commonly scalars, vectors, or matrices. The elements of a Tensor can hold integer, floating-point, or string values.

Tensor Processing Unit (TPU)

An ASIC (application-specific integrated circuit) that optimizes the performance of TensorFlow programs.

Tensor rank  
  
Tensor shape

The number of elements a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor) contains in various dimensions. For example, a [5, 10] Tensor has a shape of 5 in one dimension and 10 in another.

Tensor size

The total number of scalars a [**Tensor**](https://developers.google.com/machine-learning/glossary/#tensor) contains. For example, a [5, 10] Tensor has a size of 50.

TensorBoard  
  
TensorFlow  
  
TensorFlow Playground  
  
TensorFlow Serving  
  
test set  
  
tf.Example  
  
time series analysis

A subfield of machine learning and statistics that analyzes [**temporal data**](https://developers.google.com/machine-learning/glossary/#temporal_data). Many types of machine learning problems require time series analysis, including classification, clustering, forecasting, and anomaly detection. For example, you could use time series analysis to forecast the future sales of winter coats by month based on historical sales data.

Tokenization

the process of splitting a text string into units called tokens. A token may be a word or a group of words.

training  
  
training set  
  
transfer learning

Using a model trained to solve one problem to help to solve another problem.

For example, a *neural network* trained to distinguish between different kinds of jungle animals can be reused to train another neural network that would distinguish between different kinds of domestic animals.

translational invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the position of objects within the image changes. For example, the algorithm can still identify a dog, whether it is in the center of the frame or at the left end of the frame.

true negative (TN)

An example in which the model *correctly* predicted the [**negative class**](https://developers.google.com/machine-learning/glossary/#negative_class). For example, the model inferred that a particular email message was not spam, and that email message really was not spam.

true positive (TP)

An example in which the model *correctly* predicted the [**positive class**](https://developers.google.com/machine-learning/glossary/#positive_class). For example, the model inferred that a particular email message was spam, and that email message really was spam.

true positive rate (TP rate)

Underfitting

a situation in which the *model* trained on the *training data* **doesn't predict well *training examples***.

unlabeled example

An example that contains [**features**](https://developers.google.com/machine-learning/glossary/#feature) but no [**label**](https://developers.google.com/machine-learning/glossary/#label). Unlabeled examples are the input to [**inference**](https://developers.google.com/machine-learning/glossary/#inference). In [**semi-supervised**](https://developers.google.com/machine-learning/glossary/#semi-supervised_learning) and[**unsupervised**](https://developers.google.com/machine-learning/glossary/#unsupervised_machine_learning) learning, unlabeled examples are used during training.

unsupervised machine learning -> compare with supervised machine learning.

*Unsupervised learning* is a problem, given an *unlabeled dataset*, automatically find **hidden (or latent) structure in this dataset**.

validation loss

*Validation loss* is the average loss computed by applying the *loss function* the outputs of the model applied to the examples from the validation set.

validation set

The *validation set* is a holdout set used for *hyperparameter tuning*.

Vanishing Gradient Problem

The *vanishing gradient problem* happens in very deep neural networks, typically recurrent neural networks, that use activation functions whose gradients tend to be small. Because these small gradients are multiplied during backpropagation, they tend to "vanish" throughout the layers, preventing the network from learning long-term dependencies. Common ways to counter this problem is to use activation functions like *ReLU* or *LSTM* that do not suffer from small gradients. The opposite of this problem is called the *exploding gradient problem*.

Variance

Model contains so many noises of training model.

an error from sensitivity to small fluctuations in the *training set*. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

weight

A coefficient for a [**feature**](https://developers.google.com/machine-learning/glossary/#feature) in a linear model, or an edge in a deep network. The goal of training a linear model is to determine the ideal weight for each feature. If a weight is 0, then its corresponding feature does not contribute to the model.

wide model

A linear model that typically has many [**sparse input features**](https://developers.google.com/machine-learning/glossary/#sparse_features). We refer to it as "wide" since such a model is a special type of [**neural network**](https://developers.google.com/machine-learning/glossary/#neural_network) with a large number of inputs that connect directly to the output node. Wide models are often easier to debug and inspect than deep models. Although wide models cannot express nonlinearities through [**hidden layers**](https://developers.google.com/machine-learning/glossary/#hidden_layer), they can use transformations such as [**feature crossing**](https://developers.google.com/machine-learning/glossary/#feature_cross) and **[bucketization](https://developers.google.com/machine-learning/glossary/" \l "bucketing)** to model nonlinearities in different ways.

Contrast with [**deep model**](https://developers.google.com/machine-learning/glossary/#deep_model).

zero shot learning

Zero shot learning is the problem of learning a model capable of classifying examples whose labels weren't present in the training data. Usually, a solution involves embedding both *feature vectors* and class labels. For example, if the problem is to classify animals by their pictures, the training set can not contain an image of a dog. However, the classifier will predict the class embedding which can be used to find the corresponding label by looking in the embedding-label lookup table. The most frequently used embeddings for class labels are word2vec. However, one can train its own embedding for class labels in the case when the label can be multi-word.

Plus: (term explanation, example/deployment, use experience)

* Background

Time series

* Experience
* ARIMA, ARMA

General Data Science

* unstructured data
* association rules
* generic algorithm? Limitation
* model on the backend of project
* framework for ML
* bias vs variance
* ROC
* KNN vs Kmeans
* plotting library in Scala
* RNN, LSTM, CNN, FNN, limitation
* Deep learning models utilized
* Bidirectional LSTM and LSTM
* LSTM & GRU
* PCA & autoencoder
* Prevent overfitting with neural network
* Serverless architecture VS traditional systems
* Deployment of ML in a production environment (to prove you actually worked before) Architecture

Deployment basically means provide model you built , your prediction to a lot of people, people in your organization, general public. Live data come to your entity, and get prediction.

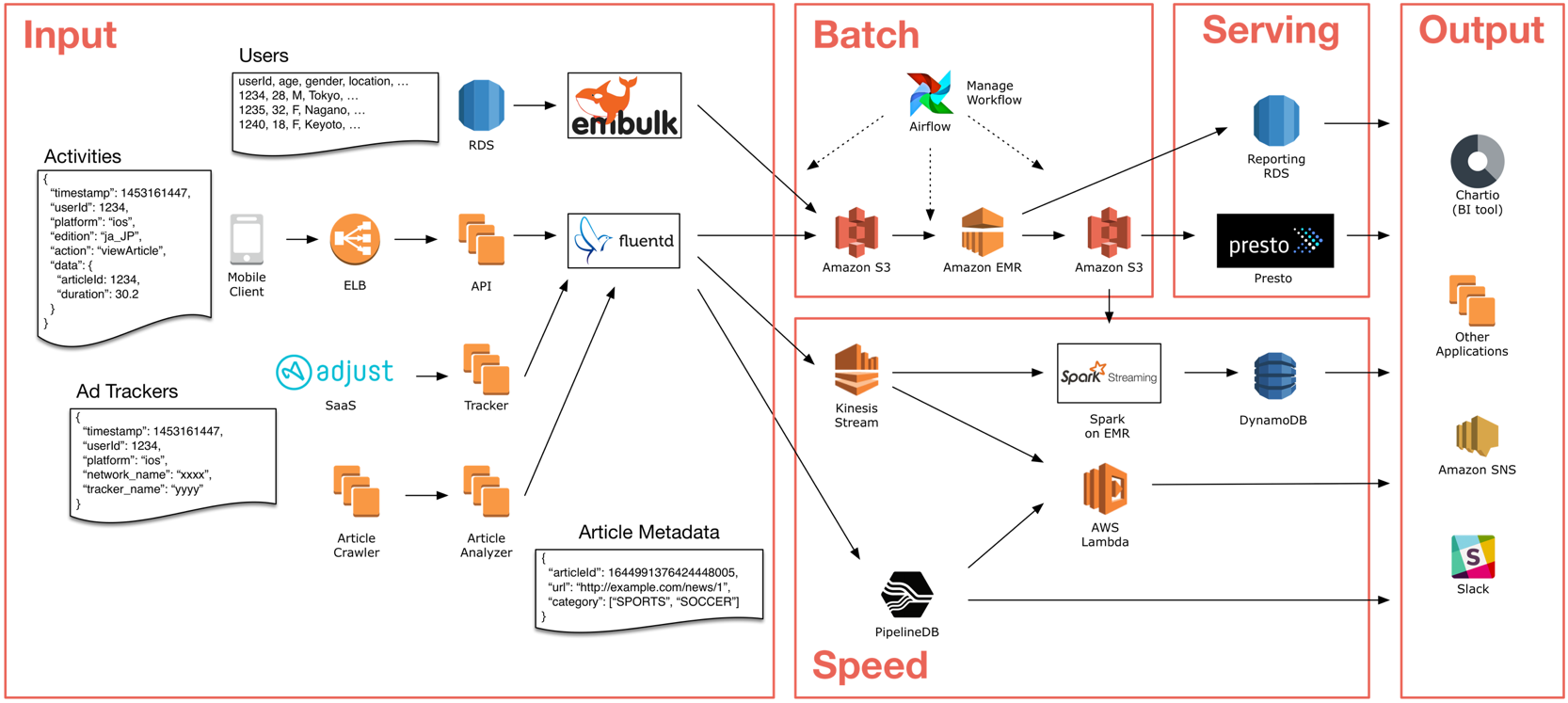
* [dplyr library](http://genomicsclass.github.io/book/pages/dplyr_tutorial.html)

Do some processing on said data, clean it up, do some aggregation and filtering in R

* + Selecting rows using filter( )
  + Arrange or re-order rows using arrange( )
  + Create new columns using mutate()
  + Create summaries of the data frame using summarise()
  + Group operations using group\_by()
  + Selecting columns using select( )
* R shiny
* R-packages: Gam and mgcv
* Union and union all
* Truncate and delete
* Show hidden directories / ls commands `ls -ad .\*`
* RJDBC

Big data & spark

* Big data: can’t completed on my laptop, even sometime you can’t load data
* Hadoop: break files into pieces and store in different slaves
  + Mapreduce
  + Hdfs: file store system,
  + Yarn: manage memory, processor
* Hadoop 2.0: introduce new second name node
  + Zookeeper can replicate/backup and controls name node
* Hadoop doent do well with small data, since it will fill with *meta data base* very soon.
* Pipeline
  + Sources: laptop, phone, input info, twitter, bank transactions, database, REST (json), SOAP (xml), whatever info
  + Ingestion:
    - flume (read logs, tracks)
    - sqoop (RDBMs, HDFS)
    - kafka (label streaming data, backup), kinesis (AWS), same work
      * kafka on-premise, kinesis on cloud
    - azure hub
  + batch processing / streaming
    - streaming: spark (connect to kafka, feed data got from kafka into micro-batches to do streaming), storm, flink (never stops, continues stream, no micro-batches), kfk streams (never stops, continues stream, no micro-batches)
    - batch: spark (RDD, DataFrame/DataSet, distributed computation, transformation create new RDD), hive, map reduce
  + storage
    - NoSql (data’s pattern changes),
    - Cassandra (first writing slow reading, sql, 六边形存储) & Hbase (first reading then writing, faster, hdfs), mongoDB, redshift (nosql, postgreSql), azure DB, Dyname DB, Impala, ELK, S3
  + visualization: kibana, vablea, flask(python),
* how to deal with growing data, retrain&redeploy
* kafka
* pipeline work at for data science, how to design, why spark?
* detail of projects
* CAP theorem
* RDD
* Spark executor
* [Tuning of spark](https://spark.apache.org/docs/latest/tuning.html)
* Narrow dependency
* Wide transformation
* Dag in spark
* Skew join
* VPC
* Spark MLlib
* Difference betwn RDS and S3
* Redshift, take too long to load data into redshift, diagnose and resolve the problem
* Prepackage work in redshift and spark
* Experience with Hbase, Cassandra, redshift, lambda
* lambda architecture (merge streaming and batch processing)



我們的儀表板是使用 [CHARTIO](https://chartio.com/)，但基本上 CHARTIO 這種儀表板服務也只是一個 Web UI，它並不會自動幫我們把使用者的存取紀錄轉成 KPI。為了理解每天人們使用 APP 的情況，我們必須自己將所有網路伺服器（Web Servers）上的使用者存取紀錄（Log Data）做一系列的處理以後，轉變成儀表板上的 KPI。

* batch processing

use Airflow, transfer input data to Apache Hive SQL

* streaming / google pub/sub

fast, latency, 1 sec or 800 ms

* batch processing(12hr/time)

day, week, month

* data partition

a practice that enables more efficient querying and data backfilling.

* data modeling

one carefully defines table schemas and data relations to capture business metrics and dimensions.

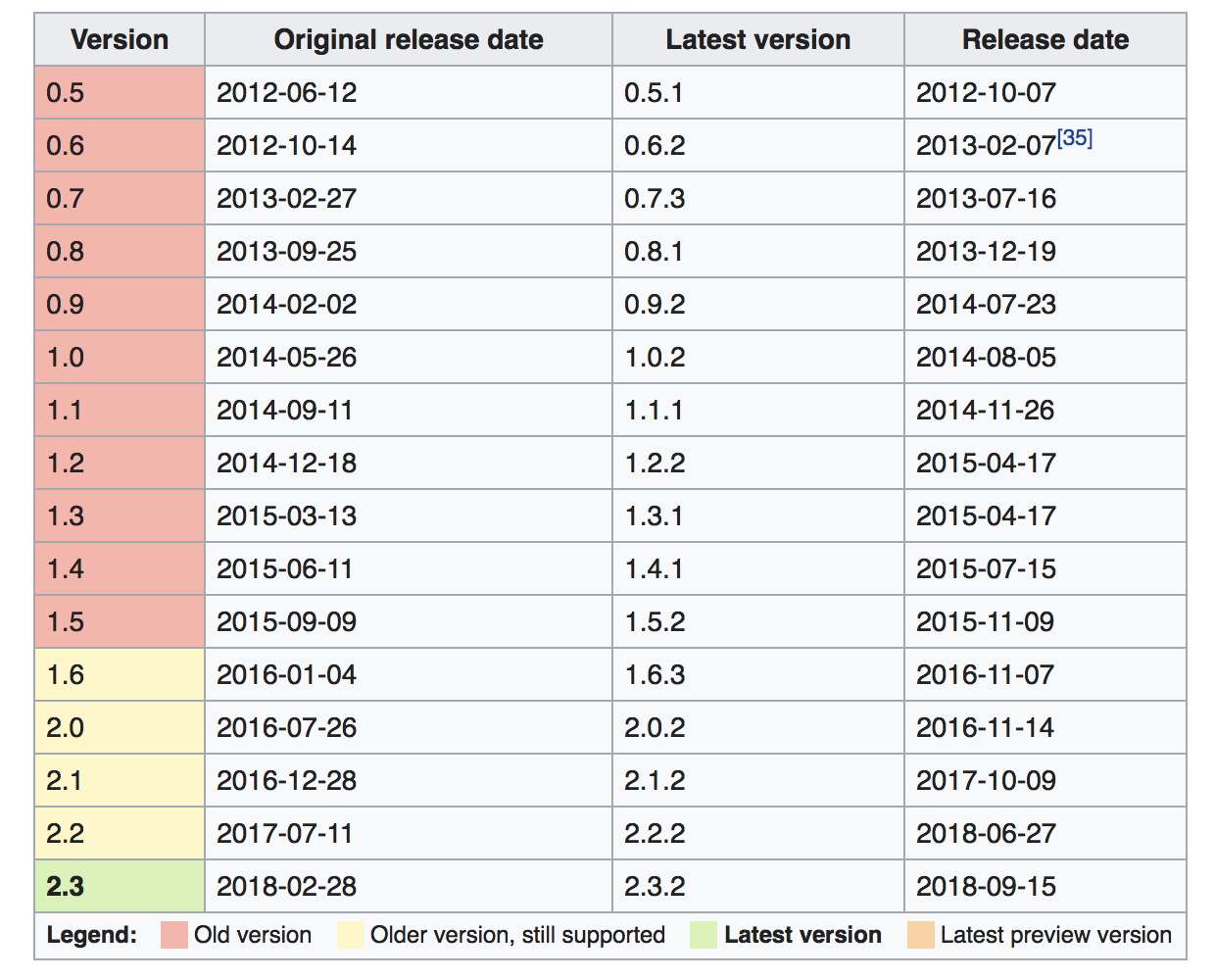
* ETL

ETL 是常用的数据处理，在以前的公司里，ETL 差不多是数据处理的基础，要求非常稳定，容错率高，而且能够很好的监控。ETL的全称是 Extract，Transform，Load， 一般情况下是将乱七八糟的数据进行预处理，然后放到储存空间上。可以是SQL的也可以是NoSQL的，还可以直接存成file的模式。

The progress of extracting data from source system and bringing it into the data warehouse.

* **Extract**is the process of *reading raw data* from a database/hardware. In this stage, the data is collected, often from multiple and different types of sources.
* **Transform** isthe process of *converting the extracted data* from its previous form into the form it needs to be in so that it can be placed into another database. Transformation occurs by using rules or lookup tables or by combining the data with other data.
* **Load** is the process of *writing the data* into the target database.



* 改變欄位名稱
* 去除空值（Missing Value）
* 套用商業邏輯，事先做資料整合（Aggregate）
* 轉變資料格式（例：從 JSON 到適合資料倉儲的格式如 [Parquet](https://parquet.apache.org/)）
* data sources
* version of spark
* tables in spark

GRU: there is a **gate gamma** tells you whether you should change your **memory cell** c, it can tell you the subject is single or plural

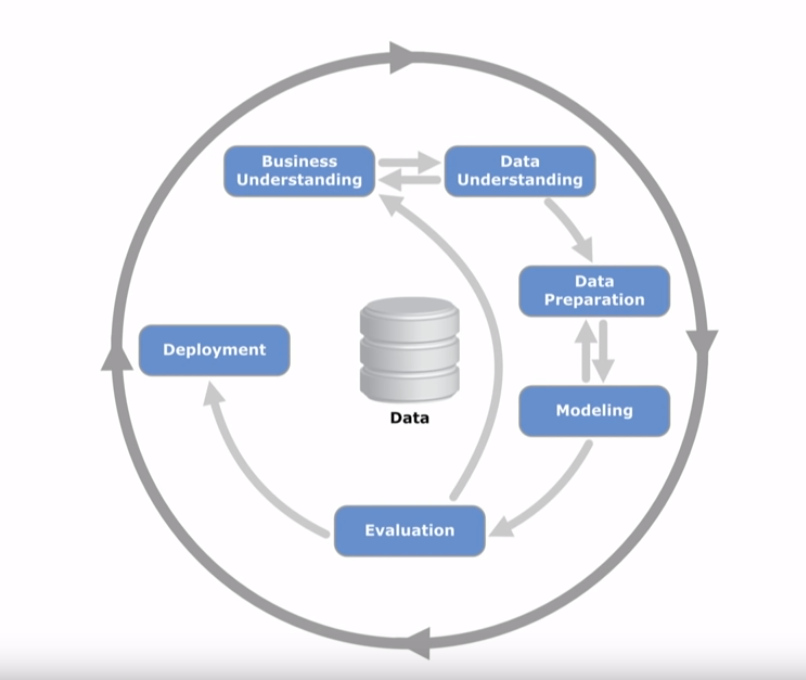
LSTM: it controls the “memorizing” process within its units using something like “gates”.

* Deployment experience in Spark, sparkml, v.s tensorflow? v.s keras?
* Apache Kafka / producer / intermediate stage
* Standardization of a schema
* Scale up? handle larger size of data
* Collaborate with ppl
* Pre-processing data in big data
* Data source
* QA environment
* Best package for a Hadoop (r?)
* Apache Drill
* Structured, semi structured and unstructured data
* productionize models
* Continuous integration
* Update model to fit new data
* Leadership (hands on / off? 自己干活么)

Engineering and architecture side

* Machine learning pipeline
* tech stack
* Zaloni
* Jenkins
* Engineering side
* machine learning flow

biz understanding → data understanding → data preparation → modeling → evaluation → **deployment**



* **Docker** experience

1. In data science, Docker is used to solve “it works on my machine” problem.

2. It sounds like virtual machine, but not exactly the same. It allow us to package and run applications in an **isolate environment, or containers**.

3. For containers we really dont’t need full scale, operating system, so they are gonna be a lot lightweight and have better performance characteristics.

4. We can use streaming or development workflows if we want to do some continuous integration or some continuous deployment, we can use docker to build micro-serveces, to do some reproducible data science.

NLP questions

* constituency parser and dependency parser
* tokenizer vs stemmers vs Lemmatizes
* NLP tools
* tells the semantic similarity of two phrases
* embedding in nlp
* glove vs word2vec
* nlp pipeline, text preprocessing
* deal with mis-spelled words in the nlp model
* how do you implement a text summarizer? Architecture? Approach? Alternatives? what would the
* training samples look like for this model? purpose of text summarizer - business use case?
* RASA
* XGboost
* NLP project?
* Part Of Speech tagging

parts-of-speech: noun, verb, pronoun, preposition, adverb, conjunction, participle, and article. Part-of-speech tagging is the process of assigning a part-of-speech marker to each word in an input text. The goal of POS-tagging is to resolve ambiguities —have more than one possible part-of-speech, choosing the proper tag for the context.

Two ways of POS:

Hidden Markov Model (HMM) – generative- given a sequence of units (words, letters, morphemes, sentences, whatever), it computes a probability distribution over possible sequences of labels and chooses the best label sequence. Markov assumption: when predicting the future, the past doesn’t matter, only the present.

Maximum Entropy Markov Model (MEMM) – discriminative-

* NER pipepline
* Text normalization

Normalizing text means converting it to a more convenient, standard form.

* + sentence segmentation

breaking up a text into individual sentences, using pucutations/cues like periods or exclamation points.

* + Tokenization

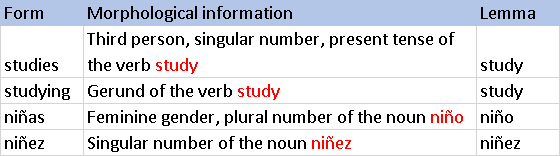
separating out or tokenizing running text into words, the task of tokenization

* + Lemmatization

Find the lemma of words. determining that two words have the same root, despite their surface differences.

E.g. sang, sung, sings $\Rightarrow$ sing(lemma 还原到原形)

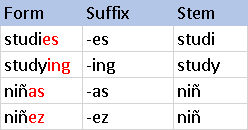
car, cars, car's, cars' $\Rightarrow$ car



* + Stemming (简单的去除affixes的词干还原, 例如cats 的词干为 cat)

a simpler version of **lemmatization** in which we mainly just strip suffixes from the end of the word. stems—the central morpheme of the word:

e.g.



* + Types

number of distinct words in a corpus

* + Token

total number N of running words.

* + named entity detection

task of detecting names, dates, and organization

* + Case folding

everything is mapped to lower case

* + Clitics

doesn’t, I’m 缩略词

* Edit distance

a way to quantify both of these intuitions about string similarity. 一个 string 要编辑多少次成为另外一个 string, 次数越少越相似

* Embedding

Vectors for representing words, word is embedded in a particular vector space, vector models of meaning are now the standard way to represent the meaning of words in NLP.

* Cosine for measuring word similarity

Cosine = v1\*v2 / |v1||v2| (v1,v2 are vectors in co-occurrence matrix)

Problem: raw frequency is very skewed and not very discriminative.

* Tf-idf algorithm for measuring word/document similarity

wt,d =tft,d×idft

* + tf: 1+log\_10 count(t,d), the frequency of the word In the document
  + idf: N/dft, where N is the total number of documents in the collection, and dft is the number of documents in which term t occurs. 出现的文档数越少越 important, 显示出 discriminative, 降低了了 the, you, we 这样的ubiquitous的词的 importance
  + collection frequency: the total number of times the word appears in the whole collection in any document.
* Embedding
* Embedding
* Embedding
* Embedding
* Embedding
* Embedding
* Embedding

Chronological knowledge

**May 2012 – Dec 2013**

**Data Scientist**

**Axiom Tech Group | Chicago, Illinois**

**Project Summary:**

Axiom provides enterprise advisory solutions and risk management. The firm relies on big data analytics to provide strategic insights and reporting to outside clients. I worked on analytics projects for various clients; cleaning data and performing analysis and reporting.

**Project Points:**

* ETL/BI requirement gathering and conversion into useful functional requirements.
* Source to target data Mapping document preparation.
* Developed report wireframes along with SQL schema data element definitions.
* Data Warehouse architecture and wrote SQL queries.
* Applied dimension modelling to identify dimension & fact tables and associated data elements.
* Familiar with wealth and asset management concepts as well as trading life cycles.
* Conducted in-depth data analysis on the reports/dashboards to identify gaps.
* Involved in data governance to find authoritative sources for the critical data elements used in the governance reports.
* Data profiling to validate data quality issues for the critical data elements.
* Participated in user acceptance testing to ensure software satisfied all requirements before it was deployed to production.
* BFSI domain and financial markets. (**Banking, Financial services and Insurance**)
* Agile process, scrum and sprint concepts.
* JIRA (issue tracking and project management)
* ETL/Reporting Tester.
* Familiar with Test plan & strategy document preparation and Test case preparation based on the requirements.
* End to end testing in DWH projects.
* Validation of ETL jobs against requirements by running through Control-M scheduler.
* Validating target tables structure, constraints against ETL requirements.
* Validating target data against source data based on ETL requirements.
* Involved in test data preparation.
* Report & Dashboard testing against target tables using SQL queries.
* Worked with module testing including defect capturing in ALM.
* Experienced with complete SDLC and STLC life cycles.
* Worked extensively with on-time delivery, process improvement, regular interaction with client and mentoring the team.

**Jan 2014 – May 2015**

**Data Scientist**

**Swift Transportation | Norfolk, Virginia**

**Project Summary:**

The transportation industry uses Big Data analytics to transform supply chain logistics -- to optimize routing, to streamline factory functions, and to give transparency to the entire supply chain. The data comes from a variety of sources, including enterprise systems, traffic analysis, weather forecasts, location information, mobile internet and GPS enabled smartphones. Logistics is a dynamic and complex process which is prone to bottlenecks, particularly in the last mile of a supply chain.

The analytics group analyzed and addressed logistics optimization. The goal was to save money and time, avoid late delivery and manage utilization of resources. This included planning routes, managing employment and fleets. Data was utilized on destination locations, shipping area, parking and time of delivery to prevent backlogs at the unloading point, and ensure deliveries were made at a time when they could be received.

**Project Points:**

* Identified and executed process improvements, hands-on in various technologies such as Oracle, Informatica, and Business Objects.
* Designed both 3NF data models for ODS, OLTP systems and dimensional data models using Star and Snow Flake Schemas.
* Developed large data sets from structured and unstructured data. Perform data mining.
* Partnered with modeling experts to develop data frame requirements for projects.
* Performed Ad-hoc reporting/customer profiling, segmentation using R/Python.
* Created statistical models for the collected data, exploratory, pre-processing, to provide conclusions with decision guides.
* Programmed a utility in Python that used multiple packages (scipy, numpy, pandas)
* Implemented Classification using supervised algorithms: Logistic Regression, Decision trees, KNN, Naive Bayes.
* Validated machine learning classifiers using ROC Curves and Lift Charts.
* Extracted data from HDFS and prepared data for exploratory analysis using data munging.
* Updated Python scripts to match training data with database stored in AWS Cloud Search and assign each document a response label for further classification.

**June 2015 – Nov 2016**

**Data Scientist**

**FleetCor | Atlanta, Georgia**

**Project Summary:**

FleetCor is an independent global provider of specialized payment products and services including fleet cards, food cards, corporate lodging discount cards and other specialized payment services for businesses throughout the world. Data Science is used in the industry for Fuel Audit and Reconciliation – to validate and audit fuel spending, while also improving driver performance. Analytics are also used to evaluate how capacity is performing in the marketplace and benchmark volume against peer groups.

Customer information (including contact information, demographics, purchase patterns, frequency of visits, communication preferences) can be derived from every driver with every transaction. Social media analytics can add a broad view of customer sentiment. Wi-Fi and beacon-enabled location data integrates data about users, vehicles, points of purchase and more. Connected cars provide data about vehicle wear and tear, route optimization and fuel efficiency, as well as IoT, which automates the collection of sensor-based and entry-based data.

My role on the project was to build predictive models and render business intelligence reports from varied data. The insights gleaned from this data were used to determine target markets and their issues, plan supply and demand of petroleum products and ways to save on costs with predictive analytics as well as reduce risk and examine strategic partnerships.

**Project Points:**

* Applied business analytics skills, integrated and prepared large, varied datasets and communicated results.
* Worked with specialized database architecture and computing environments.
* Developed analytic approaches to strategic business decisions.
* Performed analysis using predictive modeling, data/text mining, and statistical tools.
* Collaborated cross-functionally with team to develop actionable insights.
* Synthesized analytic results with business input to drive measurable change.
* Assisted in continual improvement of AWS data lake environment.
* Performed data visualization and developed presentation material using Tableau.
* Responsible for defining the key business problems to be solved while developing, maintaining relationships with stakeholders, SMEs, and cross-functional teams.
* Used Agile approaches, including Extreme Programming, Test-Driven Development, and Agile Scrum.
* Provided knowledge and understanding of current best practices and emerging trends within the analytics industry.
* Participated in product redesigns and enhancements to know how the changes will be tracked and to suggest product direction based on data patterns.
* Applied statistics and organizing large datasets of both structured and unstructured data.
* Worked with applied statistics and applied mathematics tools for performance optimization.
* Facilitated data collection to analyze document data processes, scenarios, and information flow.
* Determined data structures and their relations in supporting business objectives and provided useful data in reports.
* Promoted enterprise-wide business intelligence by enabling report access in SAS BI Portal and Tableau Server.

**Oct 2016 – Present**

**Data Scientist**

**Shell Oil | Houston, Texas**

**Project Summary:**

Predictive Analytics can enhance oil field production and cuts costs by finding optimal well settings and forecasting equipment failures and potential problems. The data spanned several years tracking oil wells in every major North American basin. The data included information on drilling and operational data from thousands of wells and hundreds of miles of low-pressure pipelines. Analysis of the data revealed critical issues with field deployed equipment.

This project built “digital twins” — computer models replicating above and below ground well behavior for artificial lift equipment. Input from sensor readings was applied to specific field issues: 1) improving plunger timing to realize well potential; 2) predicting preventive equipment maintenance to prevent failure in rods and submersible pumps; 3) reducing overuse of chemicals in wells.

**Project Points:**

* Strong experience in Software Development Life Cycle (SDLC) including Requirements Analysis, Design Specification and Testing as per cycle in both Waterfall and Agile methodologies.
* Worked in Git development environment.
* Experienced in Data Integration Validation and Data Quality controls for ETL process and Data Warehousing using MS Visual Studio, SSIS, SSAS, SSRS.
* Adept at using SAS Enterprise suite, Python, and Big Data related technologies including knowledge in Hadoop, Hive, Sqoop, Oozie, Flume, Map-Reduce
* Proficient in Predictive Modeling, Data Mining Methods, Factor Analysis, ANOVA, Hypothetical Testing, and Normal Distribution.
* Expertise in transforming business requirements into analytical models, designing algorithms, building models, developing data mining and reporting solutions that scales across massive volume of structured and unstructured data.
* Professional competency in Statistical NLP /Machine Learning, especially Supervised Learning- Document classification, information extraction, and named entity recognition in-context.
* Worked with Proof of Concepts (POC's) and gap analysis and gathered necessary data for analysis from different sources, prepared data for data exploration using data wrangling.
* Designed Physical Data Architecture of New system engines.
* Hands on experience in implementing neural network skilled in Random Forests, Decision Trees, Linear and Logistic Regression, SVM, Clustering, neural networks, Principle Component Analysis and good knowledge on Recommender Systems.
* Strong SQL Server and Python programming skills with experience in working with functions
* Efficient in developing Logical and Physical Data model and organizing data as per the business requirements using Sybase Power Designer, ER Studio in both OLTP and OLAP applications
* Experience in designing star schema, Snow flake schema for Data Warehouse, ODS architecture.
* Experience and Technical proficiency in Designing, Data Modeling Online Applications, Solution Lead for Architecting Data Warehouse/Business Intelligence Applications.
* Worked with languages like Python and Scala and software packages such as Stata, SAS and SPSS to develop neural network and cluster analysis.
* Designed visualizations using Tableau software and publishing and presenting dashboards, Storyline on web and desktop platforms.
* Developed Logical Data Architecture with adherence to Enterprise Architecture.
* Used dplyr in R and pandas in Python for performing Exploratory data analysis.
* data modeling tools like Power Designer and ER Studio.
* R and Python, Big Data technologies like Hadoop 2, HIVE, HDFS, MapReduce, and Spark.
* Spark 2.1, Spark SQL and PySpark.
* Normalization & De-Normalization techniques for optimum performance in relational and dimensional database environments.
* System Analysis, E-R/Dimensional Data Modeling, Database Design and implementing RDBMS specific features.
* Responsible for Data Analytics, Data Reporting, Ad-hoc Reporting, Graphs, Scales, PivotTables and OLAP reporting.
* Interacted with data from Hadoop for basic analysis and extraction of data in the infrastructure to provide data summarization.
* Created visualization tools and dashboards with Tableau, ggplot2 and d3.js.
* database sources like Oracle, SQL Server, and DB2.