Estimators

* [**Contents**](https://www.tensorflow.org/programmers_guide/estimators#top_of_page)
* [Advantages of Estimators](https://www.tensorflow.org/programmers_guide/estimators#advantages_of_estimators)
* [Pre-made Estimators](https://www.tensorflow.org/programmers_guide/estimators#pre_made_estimators)
  + [Structure of a pre-made Estimators program](https://www.tensorflow.org/programmers_guide/estimators#structure_of_a_pre_made_estimators_program)
  + [Benefits of pre-made Estimators](https://www.tensorflow.org/programmers_guide/estimators#benefits_of_pre_made_estimators)
* [Custom Estimators](https://www.tensorflow.org/programmers_guide/estimators#custom_estimators)
* [Recommended workflow](https://www.tensorflow.org/programmers_guide/estimators#recommended_workflow)
* [Creating Estimators from Keras models](https://www.tensorflow.org/programmers_guide/estimators#creating_estimators_from_keras_models)

This document introduces [**Estimators**](https://www.tensorflow.org/api_docs/python/tf/estimator)--a high-level TensorFlow API that greatly simplifies machine learning programming. Estimators encapsulate the following actions:

* training
* evaluation
* prediction
* export for serving

You may either use the pre-made Estimators we provide or write your own custom Estimators. All Estimators--whether pre-made or custom--are classes based on the [tf.estimator.Estimator](https://www.tensorflow.org/api_docs/python/tf/estimator/Estimator) class.

**Note:** TensorFlow also includes a deprecated **Estimator** class at [**tf.contrib.learn.Estimator**](https://www.tensorflow.org/api_docs/python/tf/contrib/learn/Estimator), which you should not use.

Advantages of Estimators

Estimators provide the following benefits:

* You can run Estimators-based models on a local host or on a distributed multi-server environment without changing your model. Furthermore, you can run Estimators-based models on CPUs, GPUs, or TPUs without recoding your model.
* Estimators simplify sharing implementations between model developers.
* You can develop a state of the art model with high-level intuitive code, In short, it is generally much easier to create models with Estimators than with the low-level TensorFlow APIs.
* Estimators are themselves built on tf.layers, which simplifies customization.
* Estimators build the graph for you. In other words, you don't have to build the graph.
* Estimators provide a safe distributed training loop that controls how and when to:
  + build the graph
  + initialize variables
  + start queues
  + handle exceptions
  + create checkpoint files and recover from failures
  + save summaries for TensorBoard

When writing an application with Estimators, you must separate the data input pipeline from the model. This separation simplifies experiments with different data sets.

Pre-made Estimators

Pre-made Estimators enable you to work at a much higher conceptual level than the base TensorFlow APIs. You no longer have to worry about creating the computational graph or sessions since Estimators handle all the "plumbing" for you. That is, pre-made Estimators create and manage [Graph](https://www.tensorflow.org/api_docs/python/tf/Graph) and [Session](https://www.tensorflow.org/api_docs/python/tf/Session) objects for you. Furthermore, pre-made Estimators let you experiment with different model architectures by making only minimal code changes.[DNNClassifier](https://www.tensorflow.org/api_docs/python/tf/estimator/DNNClassifier), for example, is a pre-made Estimator class that trains classification models through dense, feed-forward neural networks.

Structure of a pre-made Estimators program

A TensorFlow program relying on a pre-made Estimator typically consists of the following four steps:

1. **Write one or more dataset importing functions.** For example, you might create one function to import the training set and another function to import the test set. Each dataset importing function must return two objects:
   * a dictionary in which the keys are feature names and the values are Tensors (or SparseTensors) containing the corresponding feature data
   * a Tensor containing one or more labels

For example, the following code illustrates the basic skeleton for an input function:

def input\_fn(dataset):  
   ...  # manipulate dataset, extracting feature names and the label  
   return feature\_dict, label

(See [Importing Data](https://www.tensorflow.org/programmers_guide/datasets) for full details.)

1. **Define the feature columns.** Each [tf.feature\_column](https://www.tensorflow.org/api_docs/python/tf/feature_column) identifies a feature name, its type, and any input pre-processing. For example, the following snippet creates three feature columns that hold integer or floating-point data. The first two feature columns simply identify the feature's name and type. The third feature column also specifies a lambda the program will invoke to scale the raw data:

# Define three numeric feature columns.  
population = tf.feature\_column.numeric\_column('population')  
crime\_rate = tf.feature\_column.numeric\_column('crime\_rate')  
median\_education = tf.feature\_column.numeric\_column('median\_education',  
                    normalizer\_fn='lambda x: x - global\_education\_mean')

1. **Instantiate the relevant pre-made Estimator.** For example, here's a sample instantiation of a pre-made Estimator named LinearClassifier:

# Instantiate an estimator, passing the feature columns.  
estimator = tf.estimator.Estimator.LinearClassifier(  
    feature\_columns=[population, crime\_rate, median\_education],  
    )

1. **Call a training, evaluation, or inference method.** For example, all Estimators provide a train method, which trains a model.

# my\_training\_set is the function created in Step 1  
estimator.train(input\_fn=my\_training\_set, steps=2000)

Benefits of pre-made Estimators

Pre-made Estimators encode best practices, providing the following benefits:

* Best practices for determining where different parts of the computational graph should run, implementing strategies on a single machine or on a cluster.
* Best practices for event (summary) writing and universally useful summaries.

If you don't use pre-made Estimators, you must implement the preceding features yourself.

Custom Estimators

The heart of every Estimator--whether pre-made or custom--is its **model function**, which is a method that builds graphs for training, evaluation, and prediction. When you are using a pre-made Estimator, someone else has already implemented the model function. When relying on a custom Estimator, you must write the model function yourself. A[companion document](https://www.tensorflow.org/get_started/custom_estimators) explains how to write the model function.

Recommended workflow

We recommend the following workflow:

1. Assuming a suitable pre-made Estimator exists, use it to build your first model and use its results to establish a baseline.
2. Build and test your overall pipeline, including the integrity and reliability of your data with this pre-made Estimator.
3. If suitable alternative pre-made Estimators are available, run experiments to determine which pre-made Estimator produces the best results.
4. Possibly, further improve your model by building your own custom Estimator.