

## tf.estimator.regressor\_parse\_example\_spec

```
regressor_parse_example_spec(  
    feature_columns,  
    label_key,  
    label_dtype=tf.float32,  
    label_default=None,  
    label_dimension=1,  
    weight_column=None  
)
```

Defined in [tensorflow/python/estimator/canned/parsing\\_utils.py](#).

Generates parsing spec for `tf.parse_example` to be used with regressors.

If users keep data in `tf.Example` format, they need to call `tf.parse_example` with a proper feature spec. There are two main things that this utility helps:

- Users need to combine parsing spec of features with labels and weights (if any) since they are all parsed from same `tf.Example` instance. This utility combines these specs.
- It is difficult to map expected label by a regressor such as `DNNRegressor` to corresponding `tf.parse_example` spec. This utility encodes it by getting related information from users (key, dtype).

Example output of parsing spec:

```
# Define features and transformations  
feature_b = tf.feature_column.numeric_column(...)  
feature_c_bucketized = tf.feature_column.bucketized_column(  
    tf.feature_column.numeric_column("feature_c"), ...)  
feature_a_x_feature_c = tf.feature_column.crossed_column(  
    columns=["feature_a", feature_c_bucketized], ...)  
  
feature_columns = [feature_b, feature_c_bucketized, feature_a_x_feature_c]  
parsing_spec = tf.estimator.regressor_parse_example_spec(  
    feature_columns, label_key='my-label')  
  
# For the above example, regressor_parse_example_spec would return the dict:  
assert parsing_spec == {  
    "feature_a": parsing_ops.VarLenFeature(tf.string),  
    "feature_b": parsing_ops.FixedLenFeature([1], dtype=tf.float32),  
    "feature_c": parsing_ops.FixedLenFeature([1], dtype=tf.float32),  
    "my-label": parsing_ops.FixedLenFeature([1], dtype=tf.float32)  
}
```

Example usage with a regressor:

```

feature_columns = # define features via tf.feature_column
estimator = DNNRegressor(
    hidden_units=[256, 64, 16],
    feature_columns=feature_columns,
    weight_column='example-weight',
    label_dimension=3)
# This label configuration tells the regressor the following:
# * weights are retrieved with key 'example-weight'
# * label is a 3 dimension tensor with float32 dtype.

# Input builders
def input_fn_train(): # Returns a tuple of features and labels.
    features = tf.contrib.learn.read_keyed_batch_features(
        file_pattern=train_files,
        batch_size=batch_size,
        # creates parsing configuration for tf.parse_example
        features=tf.estimator.classifier_parse_example_spec(
            feature_columns,
            label_key='my-label',
            label_dimension=3,
            weight_column='example-weight'),
        reader=tf.RecordIOReader)
    labels = features.pop('my-label')
    return features, labels

estimator.train(input_fn=input_fn_train)

```

#### Args:

- `feature_columns`: An iterable containing all feature columns. All items should be instances of classes derived from `_FeatureColumn`.
- `label_key`: A string identifying the label. It means `tf.Example` stores labels with this key.
- `label_dtype`: A `tf.dtype` identifies the type of labels. By default it is `tf.float32`.
- `label_default`: used as label if `label_key` does not exist in given `tf.Example`. By default `default_value` is none, which means `tf.parse_example` will error out if there is any missing label.
- `label_dimension`: Number of regression targets per example. This is the size of the last dimension of the labels and logits `Tensor` objects (typically, these have shape `[batch_size, label_dimension]`).
- `weight_column`: A string or a `_NumericColumn` created by `tf.feature_column.numeric_column` defining feature column representing weights. It is used to down weight or boost examples during training. It will be multiplied by the loss of the example. If it is a string, it is used as a key to fetch weight tensor from the `features`. If it is a `_NumericColumn`, raw tensor is fetched by key `weight_column.key`, then `weight_column.normalizer_fn` is applied on it to get weight tensor.

#### Returns:

A dict mapping each feature key to a `FixedLenFeature` or `VarLenFeature` value.

#### Raises:

- `ValueError`: If label is used in `feature_columns`.
- `ValueError`: If `weight_column` is used in `feature_columns`.
- `ValueError`: If any of the given `feature_columns` is not a `_FeatureColumn` instance.
- `ValueError`: If `weight_column` is not a `_NumericColumn` instance.
- `ValueError`: if `label_key` is None.

---

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 3.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see our [Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated November 2, 2017.

## Stay Connected

[Blog](#)

[GitHub](#)

[Twitter](#)

## Support

[Issue Tracker](#)

[Release Notes](#)

[Stack Overflow](#)

**English**

[Terms](#) | [Privacy](#)