

Using target to generate features

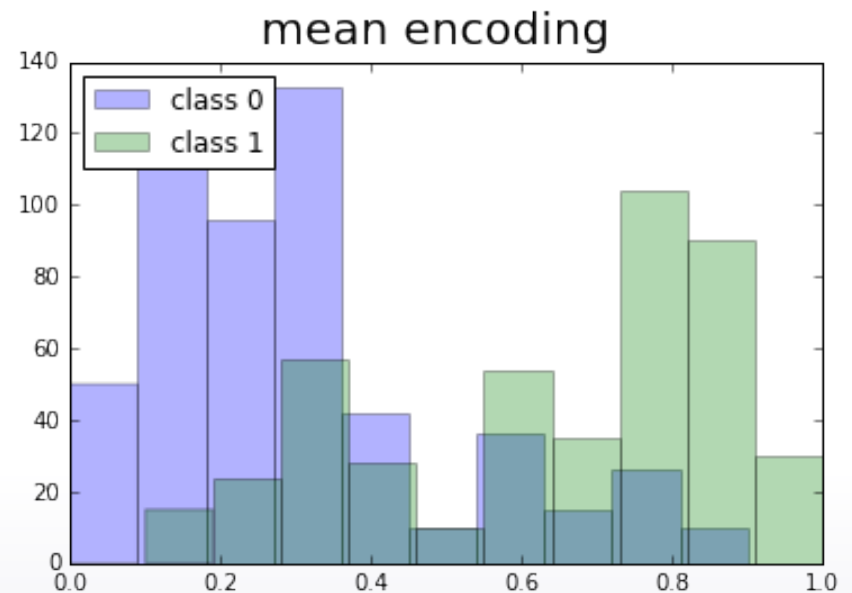
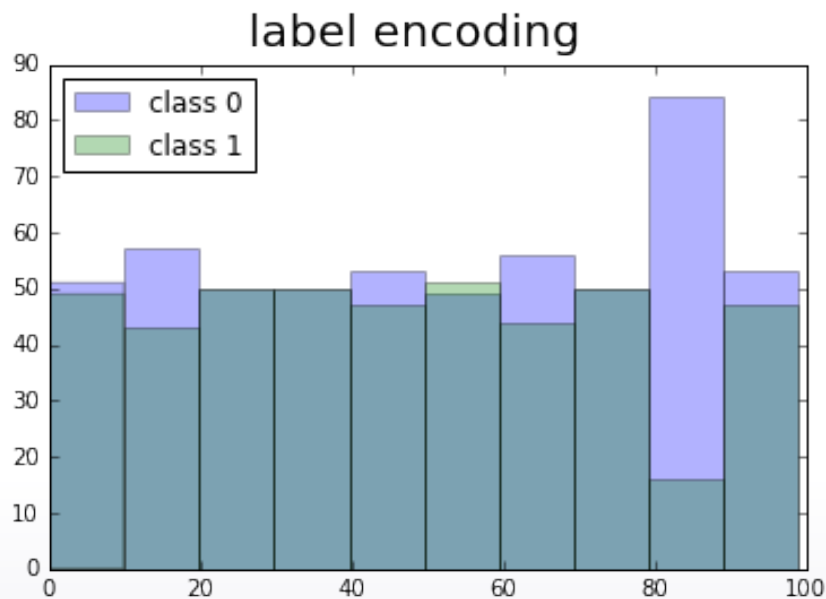
# Simple example

- Categorical feature
  - some city
- Binary classification

	feature	feature_label	feature_mean	target
<b>0</b>	Moscow	1	0.4	0
<b>1</b>	Moscow	1	0.4	1
<b>2</b>	Moscow	1	0.4	1
<b>3</b>	Moscow	1	0.4	0
<b>4</b>	Moscow	1	0.4	0
<b>5</b>	Tver	2	0.8	1
<b>6</b>	Tver	2	0.8	1
<b>7</b>	Tver	2	0.8	1
<b>8</b>	Tver	2	0.8	0
<b>9</b>	Klin	0	0.0	0
<b>10</b>	Klin	0	0.0	0
<b>11</b>	Tver	2	0.8	1

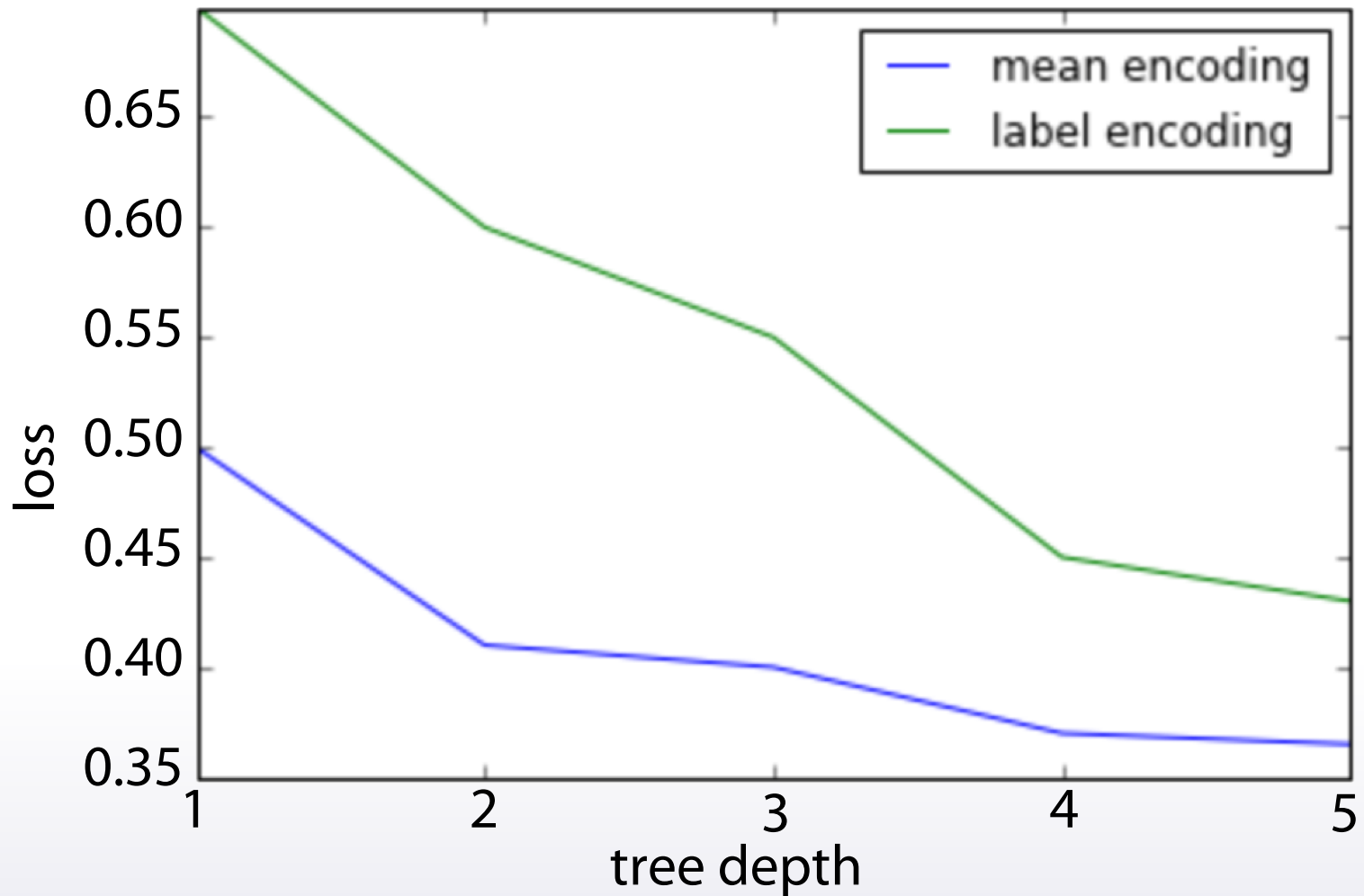
# Why does it work?

1. Label encoding gives random order. No correlation with target
2. Mean encoding helps to separate zeros from ones



# Why does it work?

Reaching a better loss with shorter trees

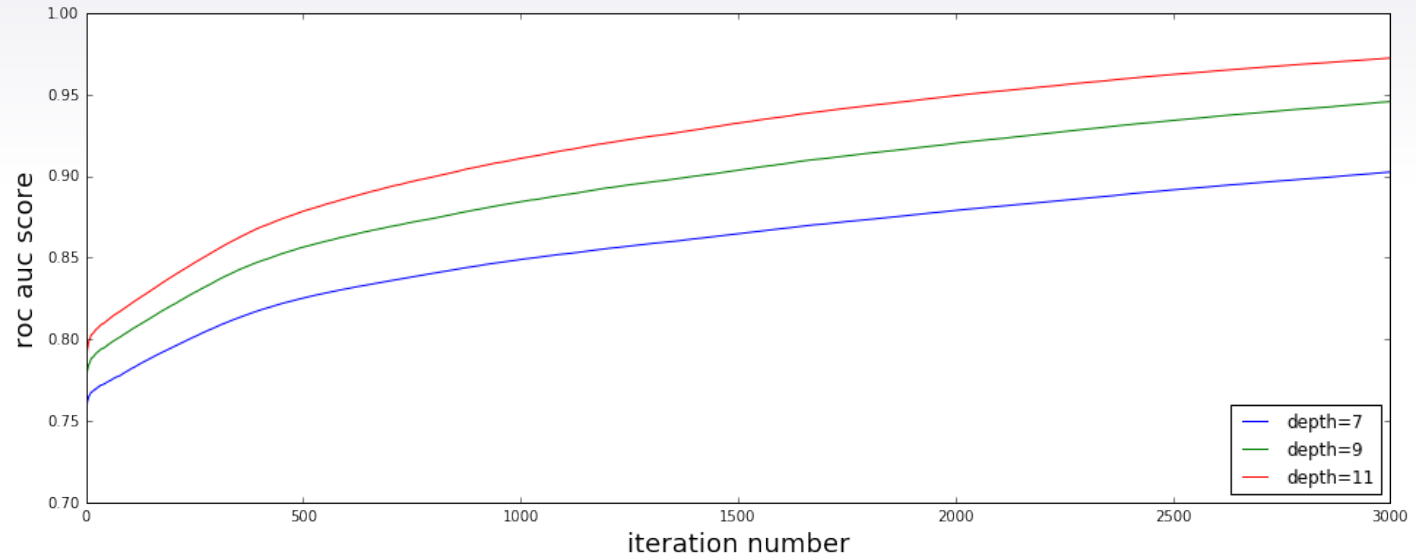


# What will you learn?

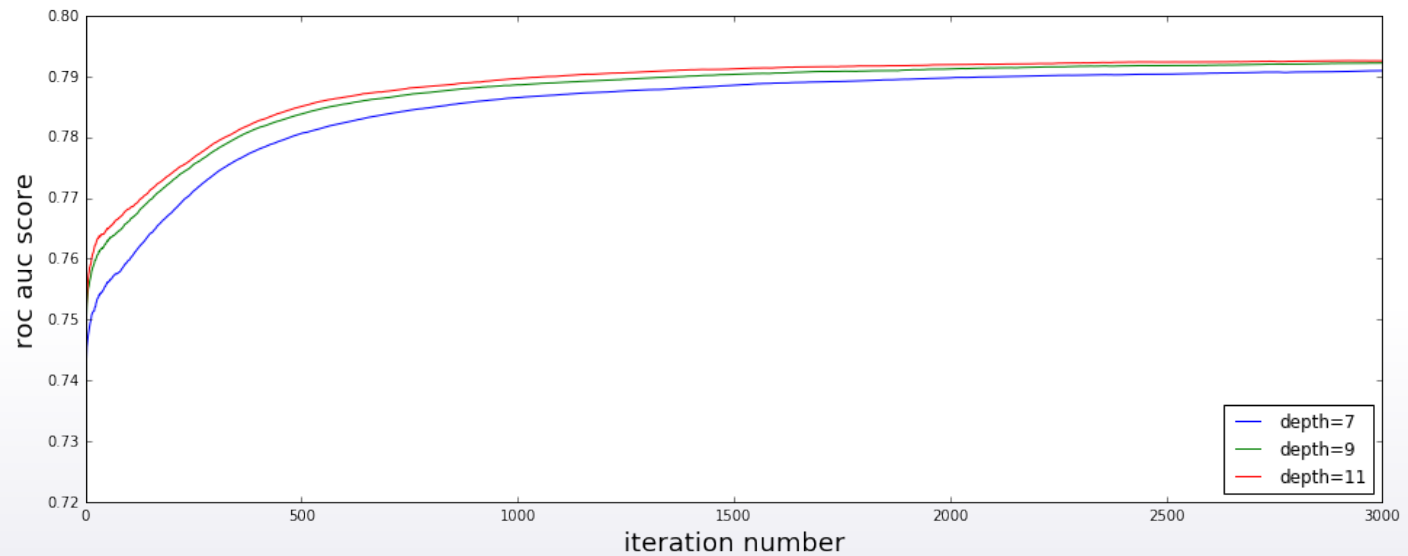
- ✓ Construct encodings
- ✓ Correctly validate them
- ✓ Extend them

# Indicators of usefulness

Train



Validation



# Ways to use target variable

Goods - number of ones in a group,

Bads - number of zeros

- $Likelihood = \frac{Goods}{Goods+Bads} = mean(target)$
- $Weight\ of\ Evidence = \ln\left(\frac{Goods}{Bads}\right) * 100$
- $Count = Goods = sum(target)$
- $Diff = Goods - Bads$

# Springleaf example

In [4]:

```
means = X_tr.groupby(col).target.mean()  
train_new[col+'_mean_target'] = train_new[col].map(means)  
val_new[col+'_mean_target'] = val_new[col].map(means)  
  
means
```

Out[4]: VAR\_1277

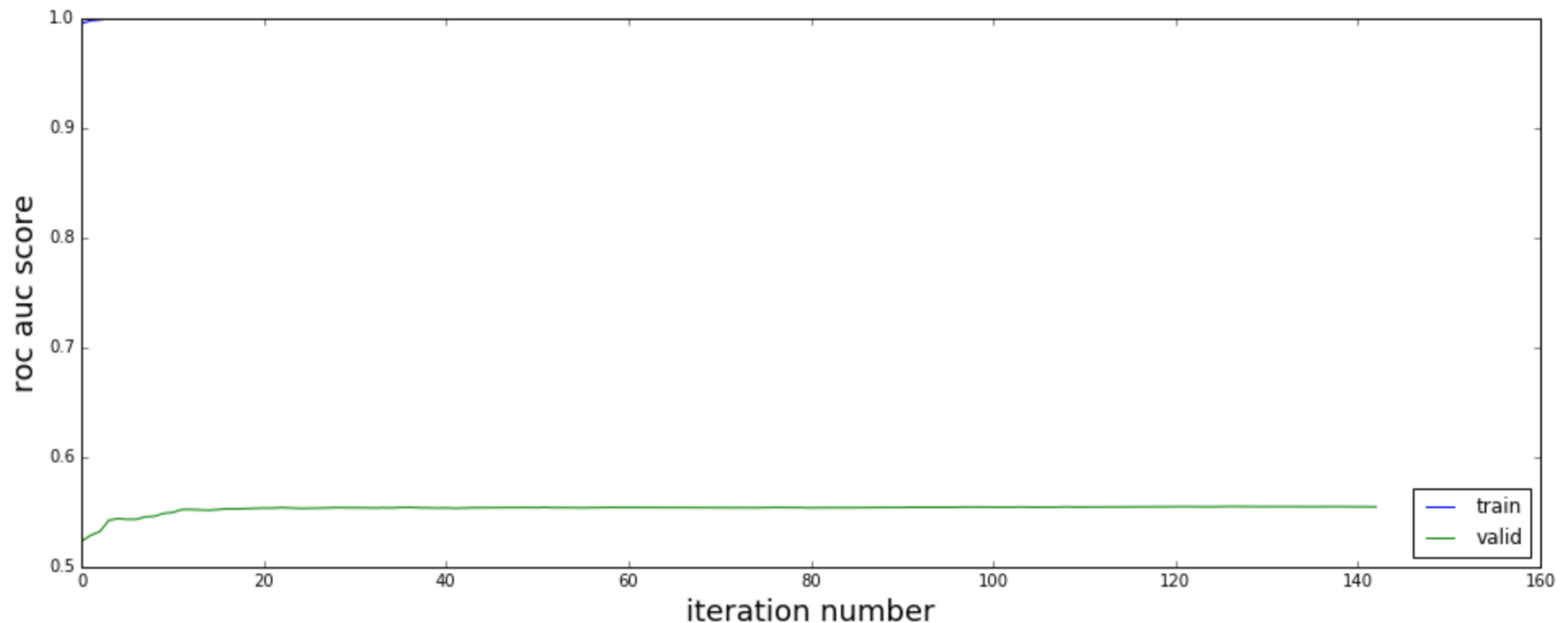
0.0	0.358965
1.0	0.219249
2.0	0.193671
3.0	0.191143
4.0	0.191080
5.0	0.185694



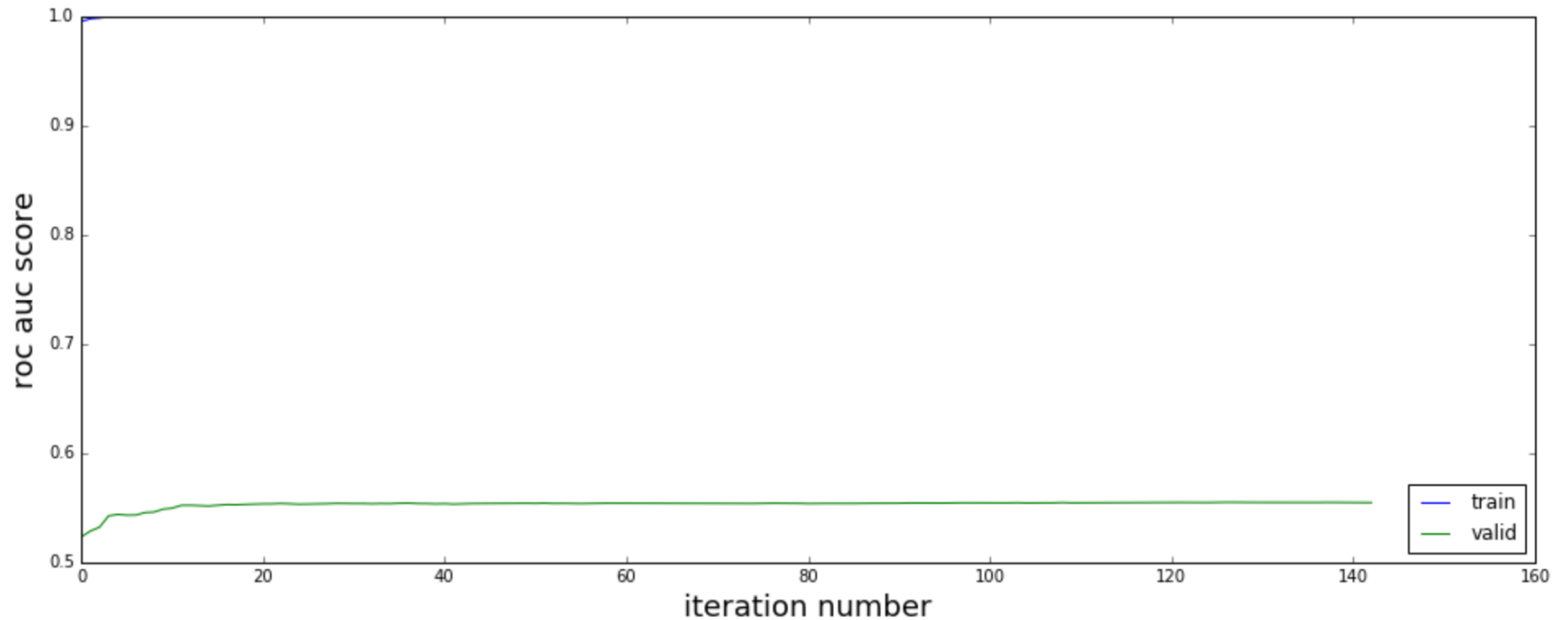
# Springleaf example

```
dtrain = xgb.DMatrix(train_new, label=y_tr)
dvalid = xgb.DMatrix(val_new, label=y_val)

evallist = [(dtrain, 'train'), (dvalid, 'eval')]
evals_result3 = {}
model = xgb.train( xgb_par, dtrain, 3000, evals=evallist,
verbose_eval=30, evals_result=evals_result3, early_stopping_rounds=50)
```



# Overfit



Train

	feature	feature_label	feature_mean	target
8	Tver	2	0	0
9	Klin	0	0	0

Validation

	feature	feature_label	feature_mean	target
10	Klin	0	1	1
11	Tver	2	1	1