Model used - custom sequential layer embeddings - 200x100 - output(2)

```
In [50]: data.show_batch(rows=1)
```

workclass	education	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	education-num_na	age	fnlwgt	education-num	target
Private	Masters	Never-married	Exec-managerial	Not-in-family	Black	Female	0	0	45	United-States	False	-0.7760	-0.3168	1.5334	<50k

```
In [31]: learn = tabular_learner(data, layers=[200,100], metrics=accuracy)

In [32]: learn.model

Out[32]: TabularModel(
           (embeds): ModuleList(
             (0): Embedding(10, 6)
             (1): Embedding(17, 8)
             (2): Embedding(8, 5)
             (3): Embedding(16, 8)
             (4): Embedding(7, 5)
             (5): Embedding(6, 4)
             (6): Embedding(3, 3)
             (7): Embedding(120, 23)
             (8): Embedding(93, 20)
             (9): Embedding(95, 20)
             (10): Embedding(43, 13)
             (11): Embedding(3, 3)
           )
           (emb_drop): Dropout(p=0.0)
           (bn_cont): BatchNorm1d(3, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
           (layers): Sequential(
             (0): Linear(in_features=121, out_features=200, bias=True)
             (1): ReLU(inplace)
             (2): BatchNorm1d(200, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
             (3): Linear(in_features=200, out_features=100, bias=True)
             (4): ReLU(inplace)
             (5): BatchNorm1d(100, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
             (6): Linear(in_features=100, out_features=2, bias=True)
           )
         )
```

Basic structures

a. It is a simple 4 layer Neural Network.
    i. Embeddings are created out of categorical inputs and contiguous inputs are taken as such.
    ii. PCA must have been used to create embeddings out of categorical data to identify the minimum numbers of features to represent a column of categorical data.
b. ReLU to introduce non-linearity
c. BatchNormalization to introduce regularization in model.
d. One more thing used for regularization is the dropout.
e. Finally output is binary classification.

Dataset:

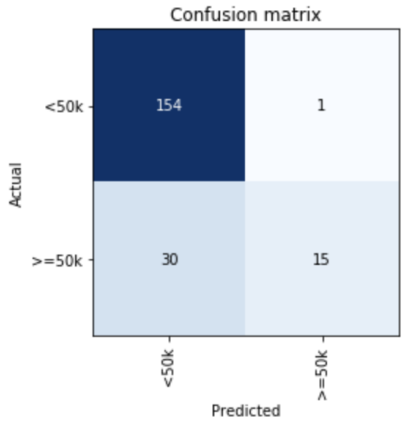
a. Fastai ADULT_SAMPLE adult.csv is used as csv dataset.
b. Categories - salary >=50k or < 50K
c. Validation set picked is a range of rows from input. This is necessary since data can be recorded in time-contiguous fashion.

Training steps-

a. Learn.lr_find() give 1e-2 as best learning rate.
b. Used it to fit the learn. Got 85% accuracy with 1 epoch.

Confusion matrix

```
In [49]: interp.plot_confusion_matrix()
```



Observation

a. Panda is used for taking in csv file.
b. Data is divided in contiguous and categorical
c. Got better accuracy with more columns as inputs.
d. Tabular data has no show_top_losses function in fastai.

Concepts explored -

a. Panda can read data from csv, hadoop and other big data stuffs.
b. There are categorical data and contiguous data for inputs.
c. Processes are performed on this tabular data just same as transforms are done on image data. These processes are like pre-processing on data to clean them up. They are FillMissing, Categorify, Normalize (contiguous data).
d. Sckit learn and boost libraries are there for machine learning on tabular data mostly but may become obsolete with deep learning tools.

Library used - fastai.tabular

Conclusion:

a. Explored a binary classification problem using deep learning technique for tabular data. Created a custom model with layers and got good accuracy.

Future work:

a. Try kaggle tabular problems.
b. Compare accuracy and try to improve it.
c. Explore epochs and learning rate.
d. Try to control embeddings created and how it affects the results.
e. How much dropout is helping.