In [33]:
```
import pandas as pd
```

In [34]:
```
df = pd.read_csv('data_category.csv')
```

In [35]:
```
df.columns
```

Out[35]:
```
Index(['acct_id', 'RPTG_SLS_MGMT_LVL_1', 'DELL_INDUSTRY_TAXONOMY_L1',
       'ORG_USER_DEFINED_VALUE_113', 'emp_size', 'buy_power', 'decision_maker',
       'time_taking'],
      dtype='object')
```

In [36]:
```
from keras.models import Sequential
from keras.layers import Dense
```

In [37]:
```
in_data = df[['RPTG_SLS_MGMT_LVL_1', 'DELL_INDUSTRY_TAXONOMY_L1',
       'ORG_USER_DEFINED_VALUE_113', 'emp_size', 'buy_power', 'decision_maker',
       'time_taking']]
df = df.fillna(0)
in_data = pd.get_dummies(in_data, columns=['RPTG_SLS_MGMT_LVL_1', 'DELL_INDUSTRY_TAXONOMY_L1',
       'ORG_USER_DEFINED_VALUE_113'])
```

In [38]:
```
in_data.columns
```

Out[38]:
```
Index(['emp_size', 'buy_power', 'decision_maker', 'time_taking',
       'RPTG_SLS_MGMT_LVL_1_APJ', 'RPTG_SLS_MGMT_LVL_1_CSB',
       'RPTG_SLS_MGMT_LVL_1_EMEA', 'RPTG_SLS_MGMT_LVL_1_Greater China',
       'RPTG_SLS_MGMT_LVL_1_LA', 'RPTG_SLS_MGMT_LVL_1_NAMERICA',
       'DELL_INDUSTRY_TAXONOMY_L1_Agriculture',
       'DELL_INDUSTRY_TAXONOMY_L1_Banking, Financial Services & Insurance',
       'DELL_INDUSTRY_TAXONOMY_L1_Business, Engineering & Consumer Services',
       'DELL_INDUSTRY_TAXONOMY_L1_Construction & Real Estate',
       'DELL_INDUSTRY_TAXONOMY_L1_Education',
       'DELL_INDUSTRY_TAXONOMY_L1_Energy',
       'DELL_INDUSTRY_TAXONOMY_L1_Government',
       'DELL_INDUSTRY_TAXONOMY_L1_Healthcare & Life Sciences',
       'DELL_INDUSTRY_TAXONOMY_L1_Manufacturing',
       'DELL_INDUSTRY_TAXONOMY_L1_Retail',
       'DELL_INDUSTRY_TAXONOMY_L1_Telecom, Media & Entertainment',
       'DELL_INDUSTRY_TAXONOMY_L1_Transportation',
       'DELL_INDUSTRY_TAXONOMY_L1_Web & Tech', 'ORG_USER_DEFINED_VALUE_113_T1',
       'ORG_USER_DEFINED_VALUE_113_T2', 'ORG_USER_DEFINED_VALUE_113_T3',
       'ORG_USER_DEFINED_VALUE_113_T4', 'ORG_USER_DEFINED_VALUE_113_T5'],
      dtype='object')
```

In [40]:
```
from sklearn.utils import shuffle
import numpy as np
in_data = shuffle(in_data)
X = in_data.drop(['time_taking'], axis = 1)
X = np.array(X)
Y = np.array(in_data['time_taking'])
```

In [41]:
```
from sklearn.preprocessing import LabelEncoder
from keras.utils.np_utils import to_categorical
l_encode = LabelEncoder()
l_encode.fit(Y)
Y = l_encode.transform(Y)
Y = to_categorical(Y)
```

In [42]:
```
Y
```

Out[42]:
```
array([[1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       ...,
       [1., 0., 0., 0.],
       [1., 0., 0., 0.],
       [1., 0., 0., 0.]], dtype=float32)
```

In [43]:
```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y = train_test_split(X,Y, test_size = 0.3, random_state = 0)
train_x.shape, train_y.shape, test_x.shape, test_y.shape
```

Out[43]:
```
((16692, 27), (16692, 4), (7155, 27), (7155, 4))
```

In [44]:
```
in_dim = len(in_data.columns)-1

model = Sequential()
model.add(Dense(8, input_dim = in_dim, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(10, activation = 'relu'))
model.add(Dense(4, activation = 'softmax'))

model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
model.fit(train_x, train_y, epochs = 15, batch_size = 5)
scores = model.evaluate(test_x, test_y)

for i, m in enumerate(model.metrics_names):
    print("\n%s: %.3f"% (m, scores[i]))
```

```
Epoch 1/15
3339/3339 [==============================] - 9s 3ms/step - loss: nan - accuracy: 0.8729
Epoch 2/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 3/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 4/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 5/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 6/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 7/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 8/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 9/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 10/15
3339/3339 [==============================] - 9s 3ms/step - loss: nan - accuracy: 0.9868
Epoch 11/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 12/15
3339/3339 [==============================] - 9s 3ms/step - loss: nan - accuracy: 0.9868
Epoch 13/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 14/15
3339/3339 [==============================] - 7s 2ms/step - loss: nan - accuracy: 0.9868
Epoch 15/15
3339/3339 [==============================] - 8s 2ms/step - loss: nan - accuracy: 0.9868
224/224 [==============================] - 1s 3ms/step - loss: nan - accuracy: 0.9877

loss: nan

accuracy: 0.988
```

In [ ]:
```

```