```
# This Python 3 environment comes with many helpful analytics libraries
installed
# It is defined by the kaggle/python docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
list the files in the input directory

import os
#####################请修改成自己存储数据的路径名
####################################
print(os.listdir("E:/input"))
print(os.listdir("E:/input/bike-sharing-dataset"))
#####################请修改成自己存储数据的路径名
####################################
# Any results you write to the current directory are saved as output.
['bike-sharing-dataset', 'data.csv', 'heart.csv', 'pima-diabetes',
'test.csv', 'train.csv']
['day.csv', 'hour.csv']
```

2

```
#############################将 hour.csv 文件读入到
raw############################################
raw = pd.read_csv("E:/input/bike-sharing-dataset/hour.csv")
#############################将 hour.csv 文件读入到
raw############################################
```

No output

Now, we are going to explore that data and understand it. The description reads as this

Both hour.csv and day.csv have the following fields, except hr which is not available in day.csv

```
- instant: record index
- dteday : date
- season : season (1:springer, 2:summer, 3:fall, 4:winter)
- yr : year (0: 2011, 1:2012)
```

```
- mnth : month ( 1 to 12)
- hr : hour (0 to 23)
- holiday : weather day is holiday or not (extracted from
http://dchr.dc.gov/page/holiday-schedule)
- weekday : day of the week
- workingday : if day is neither weekend nor holiday is 1, otherwise is 0.
+ weathersit :
    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light
Rain + Scattered clouds
    - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- temp : Normalized temperature in Celsius. The values are divided to 41
(max)
- atemp: Normalized feeling temperature in Celsius. The values are divided
to 50 (max)
- hum: Normalized humidity. The values are divided to 100 (max)
- windspeed: Normalized wind speed. The values are divided to 67 (max)
- casual: count of casual users
- registered: count of registered users
- cnt: count of total rental bikes including both casual and registered
```

3

```
####################实现功能：查看读入数据的前 5 行
####################################
raw.head()
####################实现功能：查看读入数据的前 5 行
####################################
```

3

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011	1	0	1	0	0	6	0	1	0.	0.28	0.	0.0	3	13	16

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
		/1/1									24	79	81				
1	2	2011/1/1	1	0	1	1	0	6	0	1	0.22	0.2727	0.80	0.0	8	32	40
2	3	2011/1/1	1	0	1	2	0	6	0	1	0.22	0.2727	0.80	0.0	5	27	32
3	4	2011/1/1	1	0	1	3	0	6	0	1	0.24	0.2879	0.75	0.0	3	10	13

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
		/1															
4	5	2011/1/1	1	0	1	4	0	6	0	1	0.24	0.2879	0.75	0.0	0	1	1

Lets get a deeper look

4

```
raw.describe()
```

4

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
count	17379.00000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088
std	5017.02	1.1069	0.5000	3.4387	6.9144	0.1671	2.0057	0.4654	0.6393	0.1925	0.1718	0.1929	0.1223	49.3050	151.357	181.387

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
	95	18	08	76	05	65	71	31	57	56	50	30	40	30	286	599
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000
50%	8690.00	3.0000	1.0000	7.0000	12.0000	0.0000	3.0000	1.0000	1.0000	0.5000	0.4848	0.6300	0.1940	17.0000	115.0000	142.0000

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000

Lets check the categorical variables now.

We have some variables such as the week days in which we do NOT really want to use numbers, but we just simply want to denotate whether or not a bicycle was used in a given day (Monday, Tuesday). At the moment that is done by assigning to the column "weekday" a value between 0 and 6, we want to change that... lets use dummy variables

5

```
def generate_dummies(df, dummy_column):
    #####################实现功能：将 dummy_column 中的特征转变成 one-hot 特征
###################################
    dummies = pd.get_dummies(df[dummy_column], prefix=dummy_column)
    #####################实现功能：将 dummy_column 中的特征转变成 one-hot 特征
###################################
    df = pd.concat([df, dummies], axis=1)
    return df

X = pd.DataFrame.copy(raw)
dummy_columns = ["season",     # season (1:springer, 2:summer, 3:fall,
4:winter)
              "yr",          # year (0: 2011, 1:2012)
              "mnth",        # month ( 1 to 12)
              "hr",          # hour (0 to 23)
              "weekday",     # weekday : day of the week
              "weathersit"   # weathersit :
                             # - 1: Clear, Few clouds, Partly cloudy, Partly
cloudy
                             # - 2: Mist + Cloudy, Mist + Broken clouds, Mist
+ Few clouds, Mist
                             # - 3: Light Snow, Light Rain + Thunderstorm +
Scattered clouds, Light Rain + Scattered clouds
                             # - 4: Heavy Rain + Ice Pallets + Thunderstorm +
Mist, Snow + Fog
             ]
for dummy_column in dummy_columns:
    X = generate_dummies(X, dummy_column)
```

No output

6

```
X.head()
```

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
0	1	2011/1/1	1	0	1	0	0	6	0	1	...	0	0	0	0	0	1	1	0	0	0
1	2	2011/1/1	1	0	1	1	0	6	0	1	...	0	0	0	0	0	1	1	0	0	0
2	3	2011/1/1	1	0	1	2	0	6	0	1	...	0	0	0	0	0	1	1	0	0	0

	instant	dteday	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
3	4	2011/1/1	1	0	1	3	0	6	0	1	...	0	0	0	0	0	1	1	0	0	0
4	5	2011/1/1	1	0	1	4	0	6	0	1	...	0	0	0	0	0	1	1	0	0	0

5 rows × 70 columns

7

```
X.columns
```

7

```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'hr', 'holiday',
'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt', 'season_1', 'season_2', 'season_3',
       'season_4', 'yr_0', 'yr_1', 'mnth_1', 'mnth_2', 'mnth_3', 'mnth_4',
       'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10', 'mnth_11',
```

```
       'mnth_12', 'hr_0', 'hr_1', 'hr_2', 'hr_3', 'hr_4', 'hr_5', 'hr_6',
       'hr_7', 'hr_8', 'hr_9', 'hr_10', 'hr_11', 'hr_12', 'hr_13', 'hr_14',
       'hr_15', 'hr_16', 'hr_17', 'hr_18', 'hr_19', 'hr_20', 'hr_21', 'hr_22',
       'hr_23', 'weekday_0', 'weekday_1', 'weekday_2', 'weekday_3',
       'weekday_4', 'weekday_5', 'weekday_6', 'weathersit_1', 'weathersit_2',
       'weathersit_3', 'weathersit_4'],
      dtype='object')
```

Now we need to drop the columns used originally for dummies, notice that now we have weekday_0, weekday_1 ... weekday_6, which represents Sunday to Monday (personal note here!!: I am Spanish and in Spain weekday 0 would be Monday... in English however the first day of the week is Sunday... keep in in mind!)

In any case, despite having weekday_1... weekday_6 we still have the column weekday, which is of no use already, so lets remove it along with the rest of dummy columns

8

```
for dummy_column in dummy_columns:
     ####################实现功能：删除 X 中的第 dummy_column 列
###################################
    del X[dummy_column]
    ####################实现功能：删除 X 中的第 dummy_column 列
###################################
X.columns
```

8

```
Index(['instant', 'dteday', 'holiday', 'workingday', 'temp', 'atemp', 'hum',
       'windspeed', 'casual', 'registered', 'cnt', 'season_1', 'season_2',
       'season_3', 'season_4', 'yr_0', 'yr_1', 'mnth_1', 'mnth_2', 'mnth_3',
       'mnth_4', 'mnth_5', 'mnth_6', 'mnth_7', 'mnth_8', 'mnth_9', 'mnth_10',
       'mnth_11', 'mnth_12', 'hr_0', 'hr_1', 'hr_2', 'hr_3', 'hr_4', 'hr_5',
```

```
       'hr_6', 'hr_7', 'hr_8', 'hr_9', 'hr_10', 'hr_11', 'hr_12', 'hr_13',
       'hr_14', 'hr_15', 'hr_16', 'hr_17', 'hr_18', 'hr_19', 'hr_20', 'hr_21',
       'hr_22', 'hr_23', 'weekday_0', 'weekday_1', 'weekday_2', 'weekday_3',
       'weekday_4', 'weekday_5', 'weekday_6', 'weathersit_1', 'weathersit_2',
       'weathersit_3', 'weathersit_4'],
      dtype='object')
```

And now, lets see how our data looks like

9

```
X.head()
```

9

	instant	dteday	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
0	1	2011/1/1	0	0	0.24	0.2879	0.81	0.0	3	13	...	0	0	0	0	0	1	1	0	0	0
1	2	2011/1/1	0	0	0.22	0.2727	0.80	0.0	8	32	...	0	0	0	0	0	1	1	0	0	0

	instant	dteday	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
		/1																			
2	3	2011/1/1	0	0	0.22	0.2727	0.80	0.0	5	27	...	0	0	0	0	0	1	1	0	0	0
3	4	2011/1/1	0	0	0.24	0.2879	0.75	0.0	3	10	...	0	0	0	0	0	1	1	0	0	0
4	5	2011/1	0	0	0.24	0.2879	0.75	0.0	0	1	...	0	0	0	0	0	1	1	0	0	0

	instant	dteday	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
		/1																			

5 rows × 64 columns

10

```
X.describe()
```

10

	instant	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	cnt	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
count	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	...	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00	17379.00

	instant	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	cnt	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
	00	0000	0000	0000	0000	0000	0000	0000	0000	0000		0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
mean	8690.0000	0.028770	0.682721	0.496987	0.475775	0.627229	0.190098	35.676218	153.786869	189.463088	...	0.142643	0.141147	0.142413	0.142183	0.143104	0.144542	0.656712	0.261465	0.081650	0.000173
std	5017.0295	0.167165	0.465431	0.192556	0.171850	0.192930	0.122340	49.305030	151.357286	181.387599	...	0.349719	0.348184	0.349484	0.349248	0.350189	0.351649	0.474820	0.439445	0.273839	0.013138

	instant	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	cnt	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
min	1.00000	0.000000	0.000000	0.020000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	4345.5000	0.000000	0.000000	0.340000	0.333300	0.480000	0.104500	4.000000	34.000000	40.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	8690.0000	0.000000	1.000000	0.500000	0.484800	0.630000	0.194000	17.000000	115.000000	142.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000

	instant	holiday	workingday	temp	atemp	hum	windspeed	casual	registered	cnt	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
75%	13034.5000	0.000000	1.000000	0.660000	0.621200	0.780000	0.253700	48.000000	220.000000	281.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000
max	17379.0000	1.000000	1.000000	1.000000	1.000000	1.000000	0.850700	367.000000	886.000000	977.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows × 63 columns

Time for us to plot some data and get an idea of what's going on here

11

```
first_3_weeks = 3*7*24 # 3 weeks (7 days), 24 hours each day
####################实现功能：对X中前first_3_weeks数据画图，x轴为dteday，y轴为cnt，size为（18，5）###################################
X[:first_3_weeks].plot(x='dteday', y='cnt', figsize=(18, 5))
####################实现功能：对X中前first_3_weeks数据画图，x轴为dteday，y轴为cnt，size为（18，5）###################################
```

11

```
<matplotlib.axes._subplots.AxesSubplot at 0x26f96f2e828>
```

It is also obvious that we do not need the "instant", "'dteday" columns, lets remove them

12

```
del X["instant"]
del X["dteday"]
```

No output

Finally, we need to declare which one will be our "target" column, that is, what do we want to predict? in this case it would be either "casual", "registered" or "cnt". I will use "cnt"

13

```
y = X["cnt"]
del X["cnt"]
del X["registered"]
del X["casual"]
```

No output

14

```
X.head()
```

14

	holiday	workingday	temp	atemp	hum	windspeed	season_1	season_2	season_3	season_4	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
0	0	0	0.24	0.2879	0.81	0.0	1	0	0	0	...	0	0	0	0	0	1	1	0	0	0
1	0	0	0.22	0.2727	0.80	0.0	1	0	0	0	...	0	0	0	0	0	1	1	0	0	0
2	0	0	0.22	0.2727	0.80	0.0	1	0	0	0	...	0	0	0	0	0	1	1	0	0	0
3	0	0	0.24	0.2879	0.75	0.0	1	0	0	0	...	0	0	0	0	0	1	1	0	0	0

	holiday	workingday	temp	atemp	hum	windspeed	season_1	season_2	season_3	season_4	...	weekday_1	weekday_2	weekday_3	weekday_4	weekday_5	weekday_6	weathersit_1	weathersit_2	weathersit_3	weathersit_4
4	0	0	0.24	0.2879	0.75	0.0	1	0	0	0	...	0	0	0	0	0	1	1	0	0	0

5 rows × 59 columns

We will now split into train data and test data, using 70% as train data

15

```
all_days = len(X) // 24
print("Total observations", len(X))
print("Total number of days", all_days)
days_for_training = int(len(X) * 0.7)
####################实现功能：X_train 赋值为 X 中的前 days_for_training 行数据，X_test 赋值为其余数据###################################
X_train = X[0:days_for_training]
X_test = X[days_for_training:]
####################实现功能：X_train 赋值为 X 中的前 days_for_training 行数据，X_test 赋值为其余数据###################################
```

```
Total observations 17379
Total number of days 724
```

16

```
print("Observations for training", len(X_train))
```

```
print("Observations for testing", len(X_test))
print("Some target values", y.head())
Observations for training 12165
Observations for testing 5214
Some target values 0    16
1    40
2    32
3    13
4     1
Name: cnt, dtype: int64
```

We still need to normalize our target values!

17

```
####################实现功能：将y进行最小最大标准化
##################################
y_normalized = (y - y.min()) / (y.max() - y.min())
####################实现功能：将y进行最小最大标准化
##################################
y_normalized.head()

y_train = y[0:days_for_training]
y_test = y[days_for_training:]
y_train_normalized = y_normalized[0:days_for_training]
y_test_normalized = y_normalized[days_for_training:]
```

No output

We will now build a simple model

18

```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
features = X.shape[1]
model = Sequential()
model.add(Dense(13, input_shape=(features,), activation='relu'))
model.add(Dropout(0.75))
model.add(Dense(1, activation='linear'))

model.summary()
Using TensorFlow backend.
```

```
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version
of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

```
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
D:\ProgramData\Anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
WARNING: Logging before flag parsing goes to stderr.
W1020 17:19:27.069753  9392 nn_ops.py:4224] Large dropout rate: 0.75 (>0.5).
In TensorFlow 2.x, dropout() uses dropout rate instead of keep_prob. Please
ensure that this is intended.
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)               (None, 13)                780
_________________________________________________________________
dropout_1 (Dropout)            (None, 13)                0
_________________________________________________________________
dense_2 (Dense)               (None, 1)                 14
=================================================================
Total params: 794
Trainable params: 794
Non-trainable params: 0
_________________________________________________________________
```

19

```
from keras.optimizers import SGD
sgd = SGD(lr=0.01)
model.compile(optimizer=sgd, loss="mean_squared_error")
```

No output

25

```
results = model.fit(X_train, y_train_normalized, epochs=150,validation_data
= (X_test, y_test_normalized))
Train on 12165 samples, validate on 5214 samples
Epoch 1/150
```

```
12165/12165 [==============================] - 1s 105us/step - loss: 0.0242
- val_loss: 0.0598
Epoch 2/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0242
- val_loss: 0.0589
Epoch 3/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0241
- val_loss: 0.0597
Epoch 4/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0242
- val_loss: 0.0602
Epoch 5/150
12165/12165 [==============================] - 1s 106us/step - loss: 0.0241
- val_loss: 0.0600
Epoch 6/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0240
- val_loss: 0.0591
Epoch 7/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0240
- val_loss: 0.0583
Epoch 8/150
12165/12165 [==============================] - 1s 106us/step - loss: 0.0238
- val_loss: 0.0584
Epoch 9/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0240
- val_loss: 0.0596
Epoch 10/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0240 -
val_loss: 0.0593
Epoch 11/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0241
- val_loss: 0.0589
Epoch 12/150
12165/12165 [==============================] - 1s 106us/step - loss: 0.0239
- val_loss: 0.0596
Epoch 13/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0240
- val_loss: 0.0587
Epoch 14/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0239
- val_loss: 0.0581
Epoch 15/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0238
- val_loss: 0.0587
```

```
Epoch 16/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0236 -
val_loss: 0.0585
Epoch 17/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0239
- val_loss: 0.0586
Epoch 18/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0237
- val_loss: 0.0581
Epoch 19/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0237 -
val_loss: 0.0578
Epoch 20/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0236
- val_loss: 0.0580
Epoch 21/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0236
- val_loss: 0.0589
Epoch 22/150
12165/12165 [==============================] - 1s 106us/step - loss: 0.0236
- val_loss: 0.0575
Epoch 23/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0236
- val_loss: 0.0568
Epoch 24/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0235
- val_loss: 0.0570
Epoch 25/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0236 -
val_loss: 0.0567
Epoch 26/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0234
- val_loss: 0.0568
Epoch 27/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0235
- val_loss: 0.0576
Epoch 28/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0234
- val_loss: 0.0562
Epoch 29/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0235
- val_loss: 0.0567
Epoch 30/150
```

```
12165/12165 [==============================] - 1s 101us/step - loss: 0.0231
- val_loss: 0.0559
Epoch 31/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0233
- val_loss: 0.0555
Epoch 32/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0235
- val_loss: 0.0559
Epoch 33/150
12165/12165 [==============================] - 1s 96us/step - loss: 0.0232 -
val_loss: 0.0561
Epoch 34/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0233
- val_loss: 0.0555
Epoch 35/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0230
- val_loss: 0.0547
Epoch 36/150
12165/12165 [==============================] - 1s 106us/step - loss: 0.0231
- val_loss: 0.0551
Epoch 37/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0231
- val_loss: 0.0549
Epoch 38/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0230
- val_loss: 0.0543
Epoch 39/150
12165/12165 [==============================] - 1s 107us/step - loss: 0.0232
- val_loss: 0.0547
Epoch 40/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0231
- val_loss: 0.0548
Epoch 41/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0232
- val_loss: 0.0547
Epoch 42/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0229 -
val_loss: 0.0534
Epoch 43/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0230 -
val_loss: 0.0548
Epoch 44/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0227
- val_loss: 0.0541
```

```
Epoch 45/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0229
- val_loss: 0.0538
Epoch 46/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0230
- val_loss: 0.0541
Epoch 47/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0229
- val_loss: 0.0536
Epoch 48/150
12165/12165 [==============================] - 1s 105us/step - loss: 0.0225
- val_loss: 0.0525
Epoch 49/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0227
- val_loss: 0.0534
Epoch 50/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0230 -
val_loss: 0.0529
Epoch 51/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0227 -
val_loss: 0.0528
Epoch 52/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0224 -
val_loss: 0.0521
Epoch 53/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0225 -
val_loss: 0.0527
Epoch 54/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0226
- val_loss: 0.0521
Epoch 55/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0223 -
val_loss: 0.0515
Epoch 56/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0224 -
val_loss: 0.0510
Epoch 57/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0222 -
val_loss: 0.0505
Epoch 58/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0223 -
val_loss: 0.0507
Epoch 59/150
```

```
12165/12165 [==============================] - 1s 93us/step - loss: 0.0225 -
val_loss: 0.0510
Epoch 60/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0223 -
val_loss: 0.0501
Epoch 61/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0223 -
val_loss: 0.0508
Epoch 62/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0222
- val_loss: 0.0501
Epoch 63/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0222
- val_loss: 0.0504
Epoch 64/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0226
- val_loss: 0.0503
Epoch 65/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0220
- val_loss: 0.0500
Epoch 66/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0223 -
val_loss: 0.0498
Epoch 67/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0223 -
val_loss: 0.0507
Epoch 68/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0222
- val_loss: 0.0502
Epoch 69/150
12165/12165 [==============================] - 1s 96us/step - loss: 0.0221 -
val_loss: 0.0491
Epoch 70/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0221 -
val_loss: 0.0498
Epoch 71/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0218 -
val_loss: 0.0493
Epoch 72/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0221 -
val_loss: 0.0487
Epoch 73/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0221 -
val_loss: 0.0488
```

```
Epoch 74/150
12165/12165 [==============================] - 1s 93us/step - loss: 0.0221 -
val_loss: 0.0486
Epoch 75/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0218
- val_loss: 0.0488
Epoch 76/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0219
- val_loss: 0.0482
Epoch 77/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0216
- val_loss: 0.0479
Epoch 78/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0219
- val_loss: 0.0485
Epoch 79/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0220
- val_loss: 0.0489
Epoch 80/150
12165/12165 [==============================] - 1s 115us/step - loss: 0.0217
- val_loss: 0.0473
Epoch 81/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0220
- val_loss: 0.0480
Epoch 82/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0219
- val_loss: 0.0475
Epoch 83/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0217 -
val_loss: 0.0471
Epoch 84/150
12165/12165 [==============================] - 1s 94us/step - loss: 0.0216 -
val_loss: 0.0474
Epoch 85/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0216 -
val_loss: 0.0476
Epoch 86/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0215 -
val_loss: 0.0467
Epoch 87/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0218 -
val_loss: 0.0465
Epoch 88/150
```

```
12165/12165 [==============================] - 1s 102us/step - loss: 0.0218
- val_loss: 0.0477
Epoch 89/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0215 -
val_loss: 0.0470
Epoch 90/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0468
Epoch 91/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0465
Epoch 92/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0215 -
val_loss: 0.0468
Epoch 93/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0216
- val_loss: 0.0472
Epoch 94/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0217
- val_loss: 0.0457
Epoch 95/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0215
- val_loss: 0.0457
Epoch 96/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0215 -
val_loss: 0.0465
Epoch 97/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0216 -
val_loss: 0.0467
Epoch 98/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0213
- val_loss: 0.0458
Epoch 99/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0213
- val_loss: 0.0455
Epoch 100/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0213
- val_loss: 0.0458
Epoch 101/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0214 -
val_loss: 0.0465
Epoch 102/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0214
- val_loss: 0.0463
```

```
Epoch 103/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0454
Epoch 104/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0215
- val_loss: 0.0451
Epoch 105/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0214 -
val_loss: 0.0456
Epoch 106/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0212
- val_loss: 0.0461
Epoch 107/150
12165/12165 [==============================] - 1s 105us/step - loss: 0.0212
- val_loss: 0.0455
Epoch 108/150
12165/12165 [==============================] - 1s 96us/step - loss: 0.0213 -
val_loss: 0.0458
Epoch 109/150
12165/12165 [==============================] - 1s 96us/step - loss: 0.0209 -
val_loss: 0.0460
Epoch 110/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0213
- val_loss: 0.0453
Epoch 111/150
12165/12165 [==============================] - 1s 105us/step - loss: 0.0210
- val_loss: 0.0455
Epoch 112/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0216
- val_loss: 0.0458
Epoch 113/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0211 -
val_loss: 0.0458
Epoch 114/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0211
- val_loss: 0.0453
Epoch 115/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0211 -
val_loss: 0.0444
Epoch 116/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0209
- val_loss: 0.0445
Epoch 117/150
```

```
12165/12165 [==============================] - 1s 96us/step - loss: 0.0215 -
val_loss: 0.0447
Epoch 118/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0212
- val_loss: 0.0442
Epoch 119/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0212 -
val_loss: 0.0450
Epoch 120/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0446
Epoch 121/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0213
- val_loss: 0.0450
Epoch 122/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0453
Epoch 123/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0211
- val_loss: 0.0448
Epoch 124/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0213
- val_loss: 0.0449
Epoch 125/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0213
- val_loss: 0.0457
Epoch 126/150
12165/12165 [==============================] - 1s 95us/step - loss: 0.0210 -
val_loss: 0.0449
Epoch 127/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0211 -
val_loss: 0.0442
Epoch 128/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0211
- val_loss: 0.0446
Epoch 129/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0209
- val_loss: 0.0451
Epoch 130/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0208
- val_loss: 0.0445
Epoch 131/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0210
- val_loss: 0.0442
```

```
Epoch 132/150
12165/12165 [==============================] - 1s 99us/step - loss: 0.0209 -
val_loss: 0.0444
Epoch 133/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0209 -
val_loss: 0.0448
Epoch 134/150
12165/12165 [==============================] - 1s 102us/step - loss: 0.0214
- val_loss: 0.0442
Epoch 135/150
12165/12165 [==============================] - 1s 97us/step - loss: 0.0207 -
val_loss: 0.0442
Epoch 136/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0209 -
val_loss: 0.0443
Epoch 137/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0210
- val_loss: 0.0440
Epoch 138/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0209
- val_loss: 0.0439
Epoch 139/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0212
- val_loss: 0.0440
Epoch 140/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0214
- val_loss: 0.0451
Epoch 141/150
12165/12165 [==============================] - 1s 100us/step - loss: 0.0210
- val_loss: 0.0446
Epoch 142/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0212
- val_loss: 0.0445
Epoch 143/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0213
- val_loss: 0.0438
Epoch 144/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0208
- val_loss: 0.0441
Epoch 145/150
12165/12165 [==============================] - 1s 104us/step - loss: 0.0211
- val_loss: 0.0442
Epoch 146/150
```

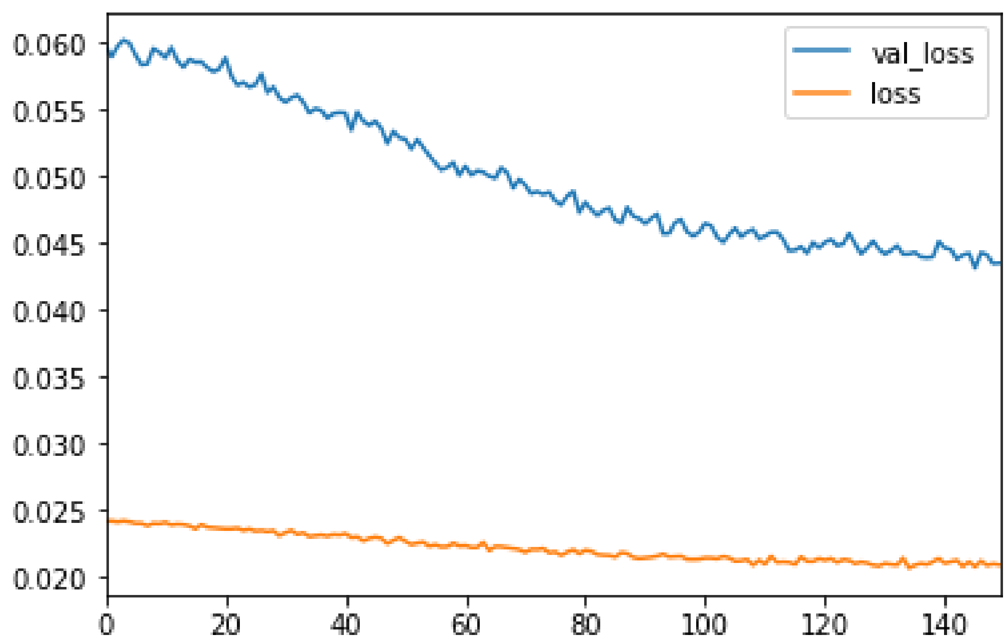
```
12165/12165 [==============================] - 1s 109us/step - loss: 0.0208
- val_loss: 0.0431
Epoch 147/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0212
- val_loss: 0.0443
Epoch 148/150
12165/12165 [==============================] - 1s 98us/step - loss: 0.0209 -
val_loss: 0.0441
Epoch 149/150
12165/12165 [==============================] - 1s 101us/step - loss: 0.0210
- val_loss: 0.0435
Epoch 150/150
12165/12165 [==============================] - 1s 103us/step - loss: 0.0209
- val_loss: 0.0435
```

26

```
results.history
pd.DataFrame.from_dict(results.history).plot()
```

26

```
<matplotlib.axes._subplots.AxesSubplot at 0x27103e9a550>
```



27

```
from sklearn.metrics import mean_squared_error, r2_score
from math import sqrt

y_test_normalized=np.array(y_test_normalized)
y_pred = model.predict(X_test)

rmse = sqrt(mean_squared_error(y_test_normalized, y_pred))
```

```
r2 = r2_score(y_test_normalized, y_pred, multioutput='raw_values')

print("RMSE:",rmse)
print("R2:",r2)

RMSE: 0.20853825503616216
R2: [0.14701181]
```