Keras_basics

Author: Srikanth KS, *gmail at sri dot teach*

This notebook gives headstart with keras library. Wine dataset is analysed, primarily on the lines of this blog post: https://www.datacamp.com/community/tutorials/deep-learning-python (https://www.datacamp.com/community/tutorials/deep-learning-python). Documented minimally, this quick and not so clean , just for later reference!

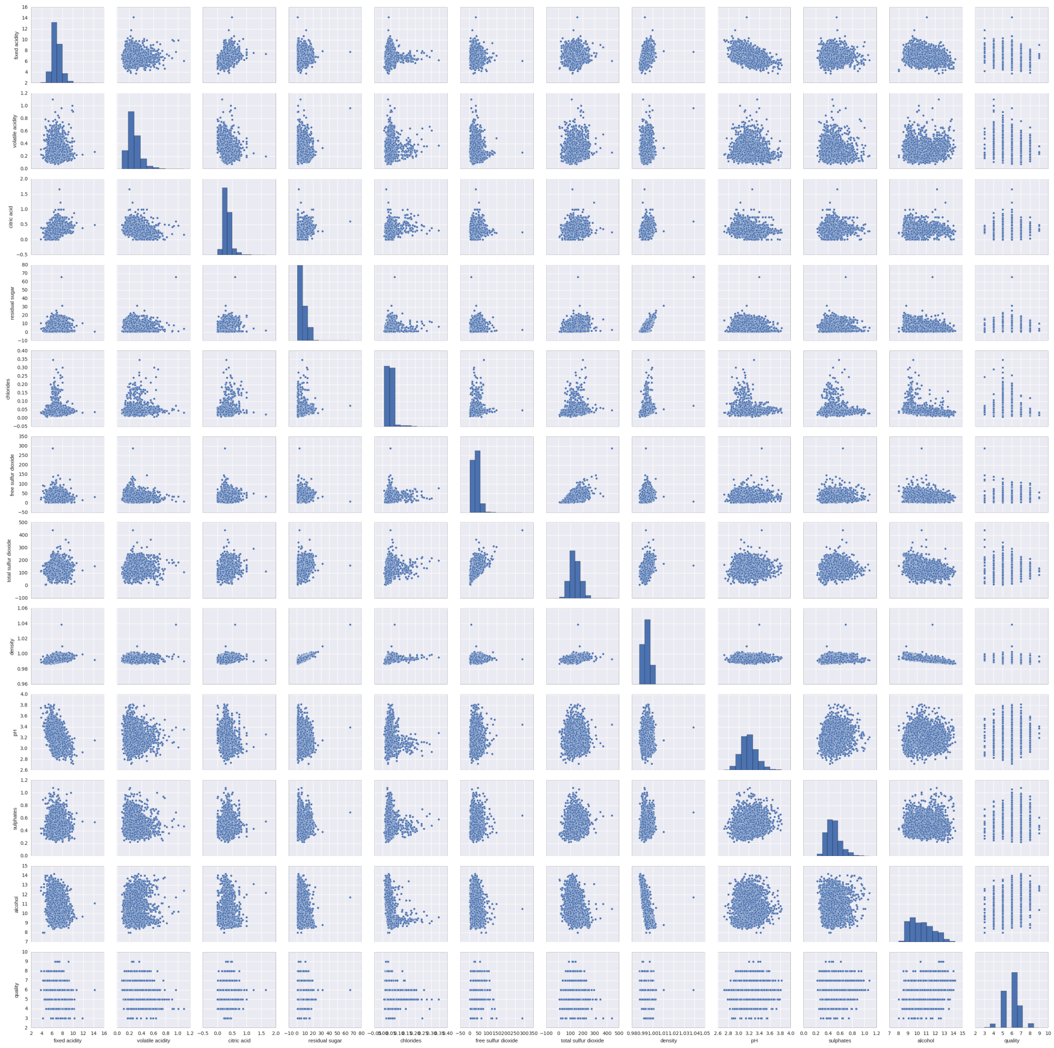
In [86]:
```
# import necessary libraries

import pandas                   as     pd
from   ggplot                   import *
import matplotlib.pyplot        as     plt
import seaborn                  as     sns
from   sklearn.model_selection  import train_test_split
from   sklearn.preprocessing    import StandardScaler
from   keras.models             import Sequential
from   keras.layers             import Dense
from   sklearn.ensemble         import RandomForestClassifier
from sklearn.metrics            import confusion_matrix
from sklearn.metrics            import precision_score
from sklearn.metrics            import recall_score
from sklearn.metrics            import f1_score
from sklearn.metrics            import cohen_kappa_score
```

In [87]:
```
# read data and summarize

baseurl = "http://archive.ics.uci.edu/ml/machine-learning-databases/wine
-quality/"
white   = pd.read_csv(baseurl + "winequality-white.csv", sep = ';')
red     = pd.read_csv(baseurl + "winequality-red.csv",   sep = ';')

print(white.info())
# print(red.info())
# print(white.describe())
# print(red.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4898 entries, 0 to 4897
Data columns (total 12 columns):
fixed acidity           4898 non-null float64
volatile acidity        4898 non-null float64
citric acid             4898 non-null float64
residual sugar          4898 non-null float64
chlorides               4898 non-null float64
free sulfur dioxide     4898 non-null float64
total sulfur dioxide    4898 non-null float64
density                 4898 non-null float64
pH                      4898 non-null float64
sulphates               4898 non-null float64
alcohol                 4898 non-null float64
quality                 4898 non-null int64
dtypes: float64(11), int64(1)
memory usage: 459.3 KB
None
```

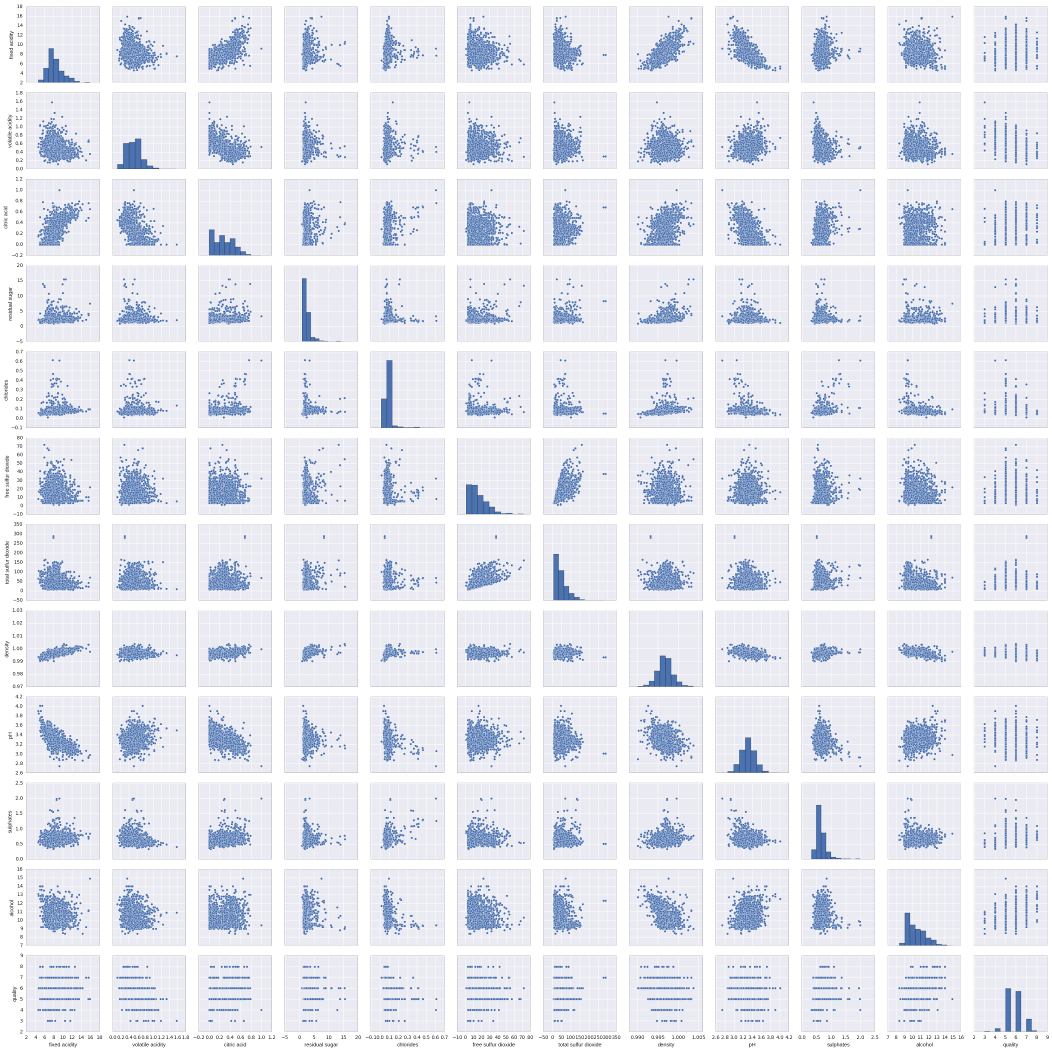
In [88]:
```
# pairwise plot for 'white'

%matplotlib inline
sns.pairplot(white)
```

Out[88]: <seaborn.axisgrid.PairGrid at 0x7f72ec649e10>



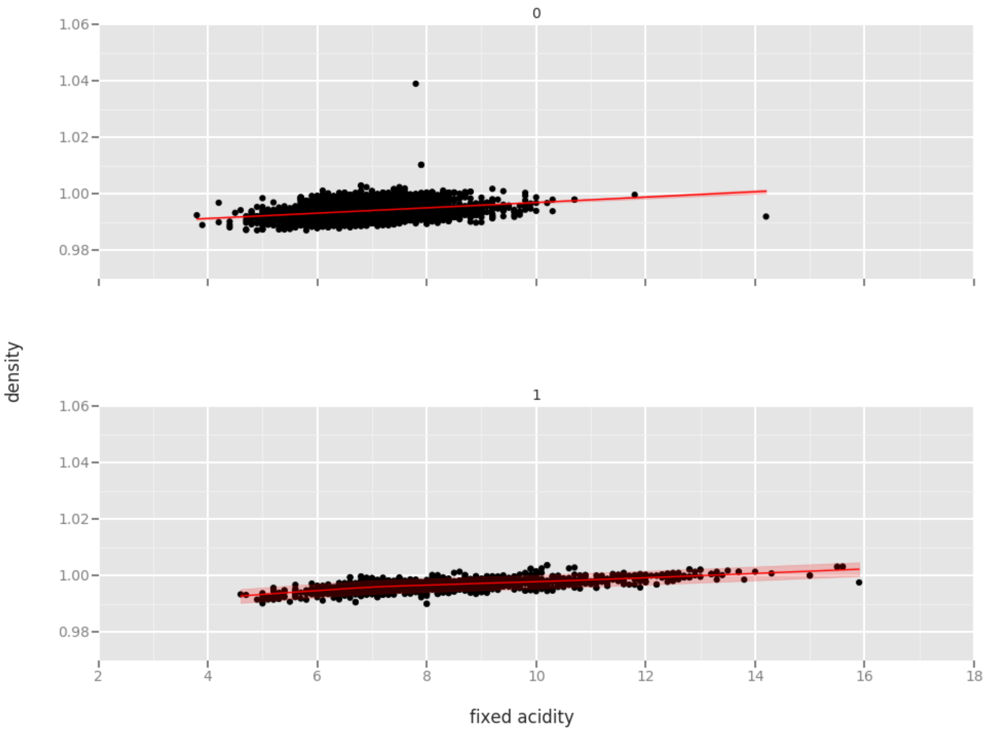
In [89]:
```
# pairwise plot for 'red'

%matplotlib inline
sns.pairplot(red)
# seems to show a predictable pattern for 'fixed acidity' versus 'densit
y' as compared to white
```

Out[89]: <seaborn.axisgrid.PairGrid at 0x7f72cef87cd0>



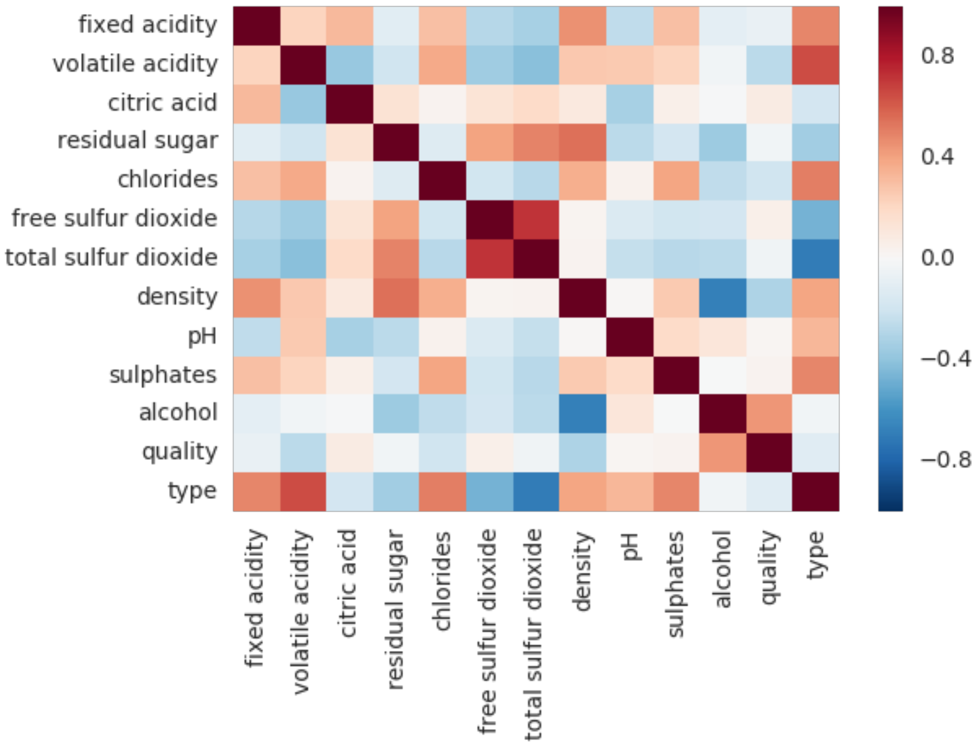
In [90]:
```
# combine datasets and visualize

red['type']   = 1
white['type'] = 0
wines         = red.append(white, ignore_index=True)

wines.info()

ggplot(aes(x='fixed acidity', y='density'), data = wines) +\
    geom_point() +\
    stat_smooth(color = 'red') +\
    facet_wrap('type')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6497 entries, 0 to 6496
Data columns (total 13 columns):
fixed acidity           6497 non-null float64
volatile acidity        6497 non-null float64
citric acid             6497 non-null float64
residual sugar          6497 non-null float64
chlorides               6497 non-null float64
free sulfur dioxide     6497 non-null float64
total sulfur dioxide    6497 non-null float64
density                 6497 non-null float64
pH                      6497 non-null float64
sulphates               6497 non-null float64
alcohol                 6497 non-null float64
quality                 6497 non-null int64
type                    6497 non-null int64
dtypes: float64(11), int64(2)
memory usage: 659.9 KB
```



Out[90]: <ggplot: (8758192696877)>

In [91]:
```
# look at correlation matrix

corr = wines.corr()
sns.heatmap(corr,
            xticklabels=corr.columns.values,
            yticklabels=corr.columns.values)
sns.plt.show()
```



In [92]:
```
# test and train split

X = wines.ix[:,0:11]
y = np.ravel(wines.type)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.
33, random_state = 42)
```

In [93]:
```
# scale the data

scaler  = StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test  = scaler.transform(X_test)
```

In [94]:
```
# set up the keras model

model = Sequential()                                                    # Initializ
e keras model
model.add(Dense(12, activation = 'relu', input_shape=(11,))) # Add an in
put layer
model.add(Dense(8, activation  = 'relu'))                              # Add one h
idden layer
model.add(Dense(1, activation  = 'sigmoid'))                           # Add an ou
tput layer
```

In [95]:

```
# look at model summary

model.summary()     # Model summary
model.get_config()  # Model config
model.get_weights() # List all weight tensors
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_4 (Dense)              (None, 12)                144       
_________________________________________________________________
dense_5 (Dense)              (None, 8)                 104       
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 9         
=================================================================
Total params: 257
Trainable params: 257
Non-trainable params: 0
_________________________________________________________________
```

```
Out[95]: [array([[ 0.03959179, -0.35063189,  0.16805166, -0.45907399, -0.15894184,
                   -0.2494964 , -0.32885724, -0.2275413 ,  0.12289608, -0.21395397,
                   -0.3073974 ,  0.04413402],
                  [-0.22700623,  0.13963783,  0.32957155,  0.39968479, -0.12256804,
                    0.16096801,  0.13777608, -0.30074504,  0.44192159, -0.31482518,
                    0.2081331 ,  0.17096585],
                  [-0.0680984 ,  0.45451027, -0.2244938 , -0.49000317, -0.27972347,
                   -0.20155856,  0.42402679, -0.43563873, -0.21070117, -0.47838563,
                   -0.49210802, -0.45533007],
                  [-0.31840932,  0.06723464,  0.23361206, -0.45476115,  0.09071827,
                   -0.43123627,  0.17900264,  0.38561523, -0.38972425,  0.48512673,
                   -0.18698582, -0.0083999 ],
                  [-0.39807361, -0.08474064, -0.1759572 ,  0.38254374, -0.10279   ,
                   -0.21508804, -0.16111621, -0.29924151,  0.20692265,  0.17008543,
                    0.26009154, -0.01738679],
                  [ 0.36916274,  0.42699951,  0.42965883,  0.18527782, -0.10988659,
                   -0.29755127,  0.24433774,  0.31619477, -0.46779478,  0.08025122,
                    0.44587642, -0.21507257],
                  [-0.3872453 ,  0.40664464,  0.04460198,  0.36493611,  0.02023602,
                    0.19201529, -0.00915515, -0.45951784,  0.50142294, -0.44824129,
                   -0.17341933,  0.06170541],
                  [-0.12345028,  0.50315922, -0.37194961, -0.2570813 ,  0.13837236,
                   -0.43090418,  0.22898954,  0.32012975, -0.40749061,  0.04169846,
                    0.18812114,  0.49456352],
                  [-0.26165915,  0.26036769,  0.27104992,  0.29298258,  0.24825674,
                   -0.20739576,  0.32487941, -0.36382455,  0.37037134, -0.38128513,
                    0.30632138, -0.11299691],
                  [ 0.35038263,  0.47433448,  0.05748683, -0.38907337,  0.02919024,
                   -0.07272491,  0.25243735, -0.37159187, -0.39948949, -0.12445298,
                    0.41566509,  0.38373482],
                  [-0.40370905, -0.12261674,  0.2495203 ,  0.13306463,  0.31399822,
                   -0.38780594, -0.2889182 ,  0.18756902, -0.22839993, -0.06953824,
                    0.10346729,  0.47586882]], dtype=float32),
          array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.], dtyp
         e=float32),
          array([[ -7.83437192e-02,   3.59107852e-01,   5.05802512e-01,
                   -5.06093204e-01,  -4.53389108e-01,  -5.04943609e-01,
                    3.22715938e-01,  -3.26121271e-01],
                  [  5.03102779e-01,   5.22499800e-01,   4.54341531e-01,
                    5.12088537e-01,  -7.52934813e-02,   3.47342610e-01,
                    9.00596380e-02,  -2.03685015e-01],
                  [ -1.38233125e-01,  -3.42139721e-01,   5.18635392e-01,
                   -2.83002645e-01,  -4.71942276e-01,   1.36966705e-01,
                    3.37976635e-01,  -4.55493897e-01],
                  [ -3.05649877e-01,   3.34713638e-01,   4.45768178e-01,
                   -3.94970238e-01,  -4.35460657e-01,   1.53878808e-01,
                   -3.09655368e-01,   5.33604741e-01],
                  [ -1.19996607e-01,  -1.07103735e-01,  -3.28565478e-01,
                    5.06619692e-01,   2.57320344e-01,  -3.67410719e-01,
                    2.43044257e-01,   2.59359956e-01],
                  [  1.97082639e-01,   2.31393814e-01,  -4.37617302e-04,
                   -1.32089108e-01,  -1.79013819e-01,  -1.12227798e-02,
                    1.13991976e-01,   1.06384456e-01],
                  [  3.76603663e-01,  -3.64697635e-01,  -4.37673450e-01,
                    1.81785405e-01,   3.18869591e-01,  -2.64533311e-01,
                   -2.81910151e-01,  -4.91049945e-01],
                  [  3.49887371e-01,   3.46854091e-01,   1.06515169e-01,
                   -1.17526293e-01,  -9.58355069e-02,  -2.02142775e-01,
                   -1.14154935e-01,  -7.91400671e-03],
                  [ -3.59570920e-01,  -6.66209161e-02,   4.41608191e-01,
                   -1.32530123e-01,   2.60664284e-01,   1.26432598e-01,
                   -2.51178682e-01,  -4.20023918e-01],
                  [  1.04037166e-01,  -3.82814527e-01,   4.17970300e-01,
                   -4.67374980e-01,  -2.53719091e-02,   3.72066498e-02,
                    5.24539709e-01,   3.43393147e-01],
                  [  1.90957963e-01,  -2.00483024e-01,  -3.27411830e-01,
                    6.90478086e-03,  -2.83585787e-02,  -3.24824274e-01,
                   -3.84428859e-01,  -3.02417994e-01],
```

In [96]:
```
# compile and fit

model.compile(loss        = 'binary_crossentropy' # for two class classi
fication
              , optimizer = 'adam'                # is a SGD variant
              , metrics   = ['accuracy'])

model.fit(X_train
          , y_train
          , epochs     = 20                       # seems to reach stabi
lity around this
          , batch_size = 1                        # send single row at a
time
          , verbose    = 1
         )
```

```
Epoch 1/20
4352/4352 [==============================] - 3s - loss: 0.0752 - acc: 0.9
729     00e+00  51/4352 [............................] - ETA: 7s - loss
: 0.5782 - acc: 0.6863
Epoch 2/20
4352/4352 [==============================] - 3s - loss: 0.0237 - acc: 0.9
956
Epoch 3/20
4352/4352 [==============================] - 3s - loss: 0.0197 - acc: 0.9
970
Epoch 4/20
4352/4352 [==============================] - 3s - loss: 0.0193 - acc: 0.9
963
Epoch 5/20
4352/4352 [==============================] - 3s - loss: 0.0159 - acc: 0.9
975
Epoch 6/20
4352/4352 [==============================] - 3s - loss: 0.0161 - acc: 0.9
972
Epoch 7/20
4352/4352 [==============================] - 3s - loss: 0.0138 - acc: 0.9
970
Epoch 8/20
4352/4352 [==============================] - 3s - loss: 0.0148 - acc: 0.9
966
Epoch 9/20
4352/4352 [==============================] - 3s - loss: 0.0130 - acc: 0.9
972
Epoch 10/20
4352/4352 [==============================] - 3s - loss: 0.0124 - acc: 0.9
975
Epoch 11/20
4352/4352 [==============================] - 3s - loss: 0.0126 - acc: 0.9
972
Epoch 12/20
4352/4352 [==============================] - 3s - loss: 0.0151 - acc: 0.9
970
Epoch 13/20
4352/4352 [==============================] - 3s - loss: 0.0108 - acc: 0.9
979
Epoch 14/20
4352/4352 [==============================] - 3s - loss: 0.0122 - acc: 0.9
979
Epoch 15/20
4352/4352 [==============================] - 3s - loss: 0.0103 - acc: 0.9
977
Epoch 16/20
4352/4352 [==============================] - 3s - loss: 0.0113 - acc: 0.9
972
Epoch 17/20
4352/4352 [==============================] - 3s - loss: 0.0101 - acc: 0.9
979
Epoch 18/20
4352/4352 [==============================] - 3s - loss: 0.0098 - acc: 0.9
979
Epoch 19/20
4352/4352 [==============================] - 3s - loss: 0.0096 - acc: 0.9
977
Epoch 20/20
4352/4352 [==============================] - 3s - loss: 0.0095 - acc: 0.9
979
```

```
Out[96]: <keras.callbacks.History at 0x7f72b7e08a50>
```

In [97]:
```
# predict

y_pred = np.round(model.predict(X_test))
print(y_pred)
```

```
[[ 0.]
 [ 1.]
 [ 0.]
 ...,
 [ 0.]
 [ 0.]
 [ 0.]]
```

In [98]:
```
# compare actual versus test to obtain 'loss' and 'accuracy'

score = model.evaluate(X_test, y_test,verbose = 1)
print("\n")
print(["loss", "accuracy"])
print(score)
```

```
  32/2145 [..............................] - ETA: 1s

['loss', 'accuracy']
[0.028386010442710361, 0.99533799533799538]
```

In [99]:
```
# Confusion matrix
print("\nConfusion matrix")
print( confusion_matrix(y_test, y_pred) )

# Precision
print("\nPrecision")
print( precision_score(y_test, y_pred) )
0.994565217391

# Recall
print("\nRecall")
print( recall_score(y_test, y_pred) )
0.98563734290843807

# F1 score
print("\nF1 score")
print( f1_score(y_test,y_pred) )
0.99008115419296661

# Cohen's kappa
print("\nCohen's kappa")
print( cohen_kappa_score(y_test, y_pred) )
```

```
Confusion matrix
[[1586    2]
 [   8  549]]

Precision
0.996370235935

Recall
0.985637342908

F1 score
0.990974729242

Cohen's kappa
0.987832175925
```

In [100]:
```
# lets see how it compares with randomforest

rf_model = RandomForestClassifier(n_estimators   = 500   # number of tre
es
                                  , verbose      = 1
                                  , oob_score    = True
                                  , random_state = 1
                                 )
rf_model.fit(X_train, y_train)
print("\noob score")
print(rf_model.oob_score_)

print("\ntest score")
print(rf_model.score(X_test, y_test))
```

```
[Parallel(n_jobs=1)]: Done 500 out of 500 | elapsed:    1.7s finished

oob score
0.993795955882

test score
0.994871794872

[Parallel(n_jobs=1)]: Done 500 out of 500 | elapsed:    0.3s finished
```

A lot of things can be done further, but that is for a different day!