Pre?sentation

October 6, 2017

In [1]:
```
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns; sns.set()

%matplotlib inline
import mpld3
mpld3.enable_notebook()
```

1 Formatage des données

1.1 1ère approche: 15 min sampling

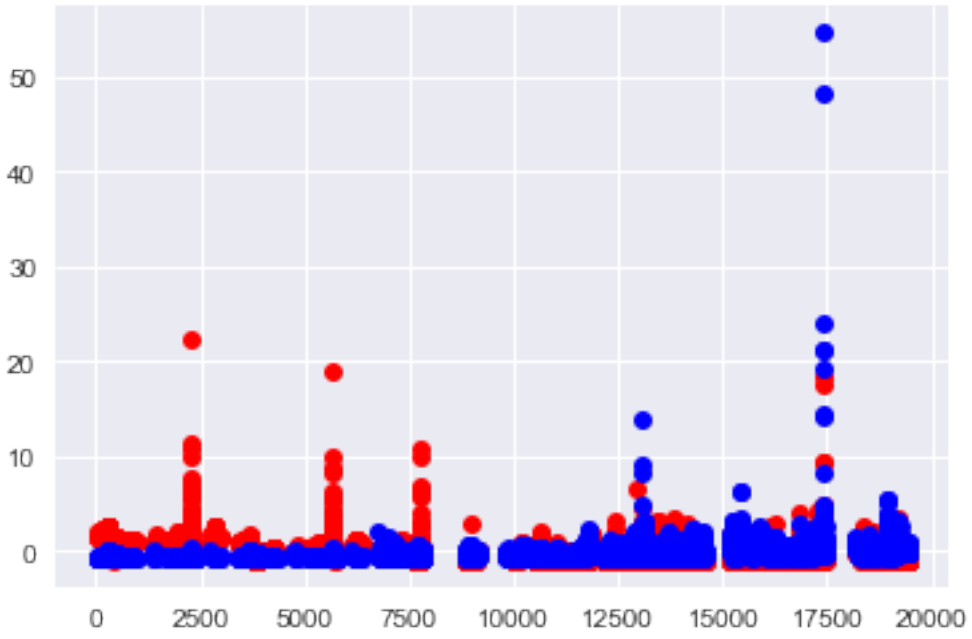
In [2]:
```
df_15_min = pd.read_pickle('data/data_15min.pkl')
df_15_min = df_15_min[pd.notnull(df_15_min).all(axis=1)]
print(len(df_15_min))
df_15_min.head()
```

```
8676
```

Out[2]:

	date	h2s	pressure	temperature	humidity	n_points	so2	h2s_ref	captor
4	2016-05-23 09:15:00	1.673433	0.047316	0.996046	-1.610853	15	-0.259294	-0.709240	1303
9	2016-05-23 10:30:00	1.530588	0.202130	0.996046	-1.677224	15	-0.259294	-0.709240	1303
20	2016-05-23 13:15:00	1.637722	0.356944	0.996046	-1.809965	15	-0.259294	-0.709240	1303
25	2016-05-23 14:30:00	1.655577	0.356944	0.996046	-1.809965	15	-0.259294	-0.561424	1303
31	2016-05-23 16:00:00	1.566299	0.356944	0.996046	-1.876336	15	-0.259294	-0.709240	1303

In [3]:
```
plt.plot(df_15_min['h2s'], 'ro')
plt.plot(df_15_min['h2s_ref'], 'bo')
plt.show()
```



1.2 2ème approche: 15 min/ref - 1 min/carpol

```
In [4]: df_1_min = pd.read_pickle('data/data_1min.pkl')
        df_1_min = df_1_min[pd.notnull(df_1_min).all(axis=1)]
        print(len(df_1_min))
        df_1_min.head()
        print(df_1_min.shape)
```

```
8651
(8651, 23)
```

2 Support Vector Regression

```
In [5]: from sklearn import svm
```

2.1 15 min sampling

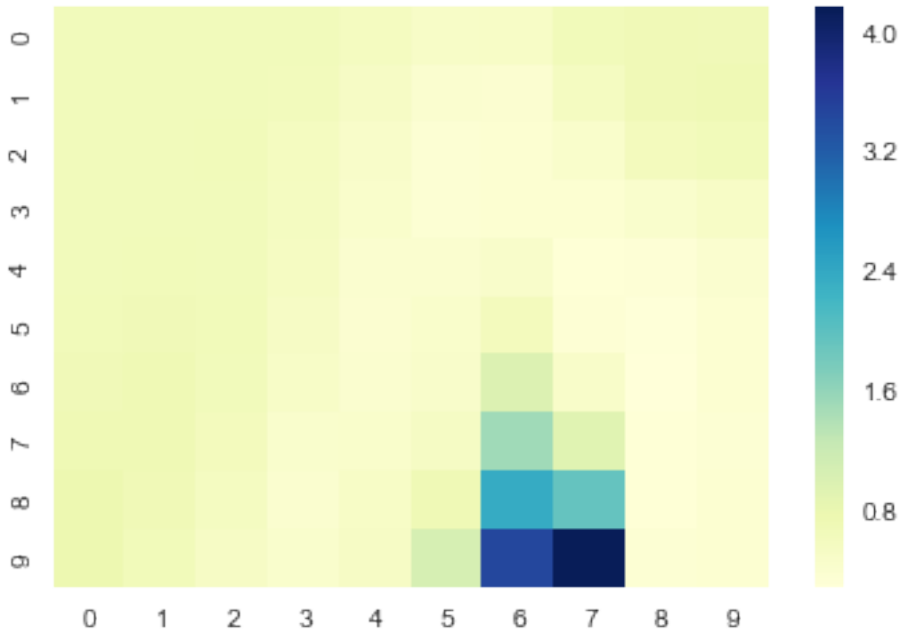
```
In [6]: # Préparation des données
        def split_dataframe(dataframe, percent):
            nb_rows = int(np.floor(percent * len(dataframe)))
            return dataframe[:nb_rows], dataframe[nb_rows:]

        df_train, df_test = split_dataframe(df_15_min, 0.5)
        df_valid, df_test = split_dataframe(df_test, 0.5)
```

```
        def dataframe_to_xy(df):
            return np.array(df[['h2s', 'pressure', 'temperature', 'humidity', 'so2']]), np.arra

        X_train_15, y_train_15 = dataframe_to_xy(df_train)
        X_valid_15, y_valid_15 = dataframe_to_xy(df_valid)
        X_test_15, y_test_15 = dataframe_to_xy(df_test)
```

In [7]:
```
        print("SVR 15 min")
        C_grid = np.logspace(-2, 3, 10)
        gamma_grid = np.logspace(-5, 2, 10)

        scores = np.empty((len(C_grid), len(gamma_grid)))
        for i in range(len(C_grid)):
            for j in range(len(gamma_grid)):
                clf = svm.SVR(C=C_grid[i], gamma=gamma_grid[j], verbose=0)
                clf.fit(X_train_15, y_train_15)
                score = clf.score(X_valid_15, y_valid_15)
                scores[i, j] = score
```

```
SVR 15 min
```

In [8]:
```
ax = sns.heatmap(-scores, cmap="YlGnBu")
```



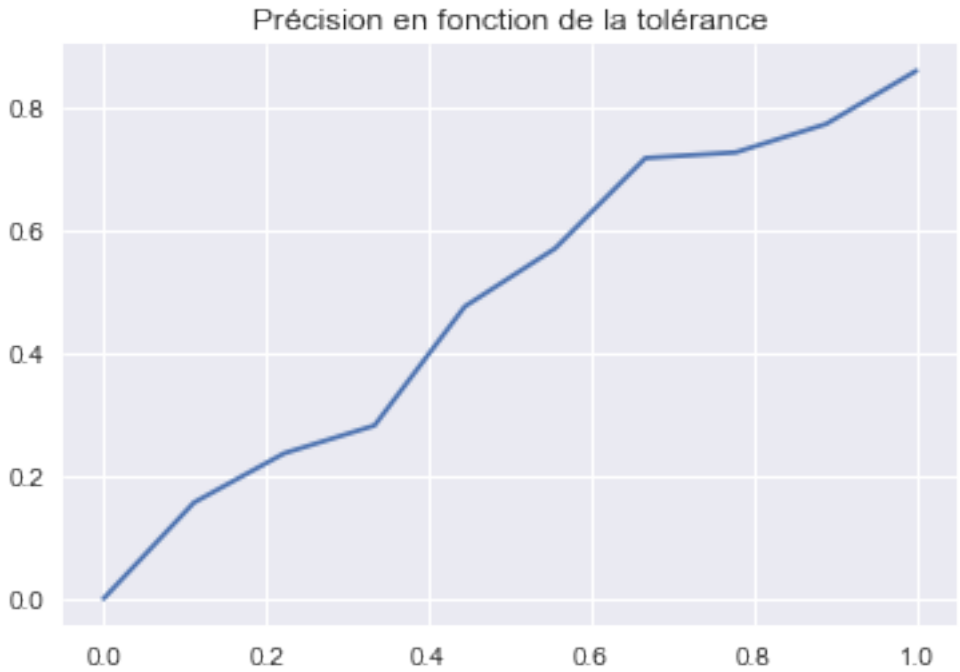
In [9]:
```
        C_idx, gamma_idx = np.unravel_index(scores.argmax(), scores.shape) # Meilleur résultat
        clf = svm.SVR(C=C_grid[C_idx], gamma=gamma_grid[gamma_idx])
```

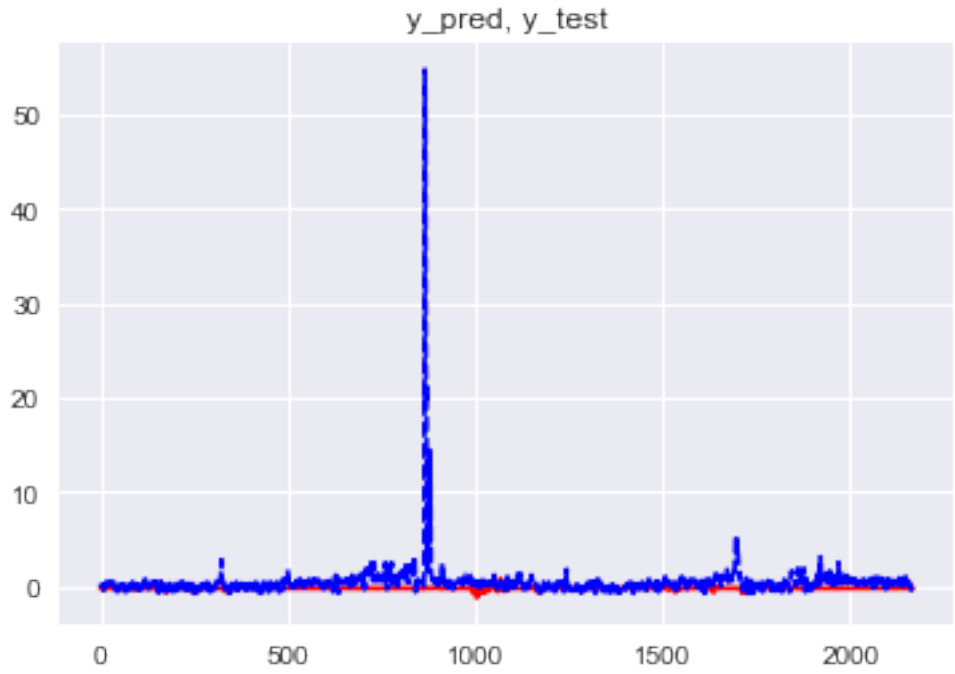
```
clf.fit(X_train_15, y_train_15)
err_test = clf.score(X_test_15, y_test_15)
```

In [10]:
```
%%latex
$$ acc = \sum_{i=0}^{n} \frac{x_{i}}{n} avec \begin{cases} x_{i} = 1: |\hat{y_{i}} - y
                                                                0: sinon \\ \end{ca
```

$$acc = \sum_{i=0}^{n} \frac{x_i}{n} avec \begin{cases} x_i = 1 : |\hat{y}_i - y_i| <= tolerance \\ 0 : sinon \end{cases}$$

In [11]:
```
y_pred = clf.predict(X_test_15)
tolerances = np.linspace(0, 1, 10)
acc = []
for tol in tolerances:
    acc.append(np.sum(np.abs(y_pred.flatten() - y_test_15.flatten()) <= tol) / len(y_t

plt.figure()
plt.title('Précision en fonction de la tolérance')
plt.plot(tolerances, acc)
plt.show()

plt.figure()
plt.title('y_pred, y_test')
plt.plot(y_pred, '-r')
plt.plot(y_test_15, '--b')
plt.show()
```





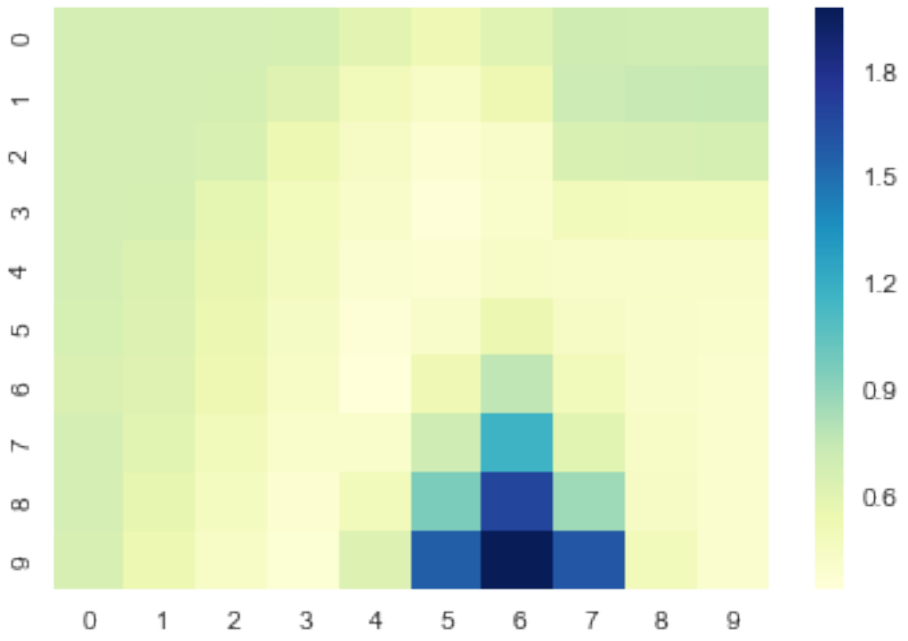
2.2 1 min sampling

```
In [12]: df_train, df_test = split_dataframe(df_1_min, 0.5)
         df_valid, df_test = split_dataframe(df_test, 0.5)

         def dataframe_to_xy(df):
             return np.array(df[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 'pressure',

         X_train_1, y_train_1 = dataframe_to_xy(df_train)
         X_valid_1, y_valid_1 = dataframe_to_xy(df_valid)
         X_test_1, y_test_1 = dataframe_to_xy(df_test)
```

```
In [13]: print("SVR 1 MIN")
         C_grid = np.logspace(-2, 3, 10)
         gamma_grid = np.logspace(-5, 2, 10)

         scores = np.empty((len(C_grid), len(gamma_grid)))
         for i in range(len(C_grid)):
             for j in range(len(gamma_grid)):
                 clf = svm.SVR(C=C_grid[i], gamma=gamma_grid[j], verbose=0)
                 clf.fit(X_train_1, y_train_1)
                 score = clf.score(X_valid_1, y_valid_1)
                 scores[i, j] = score
```

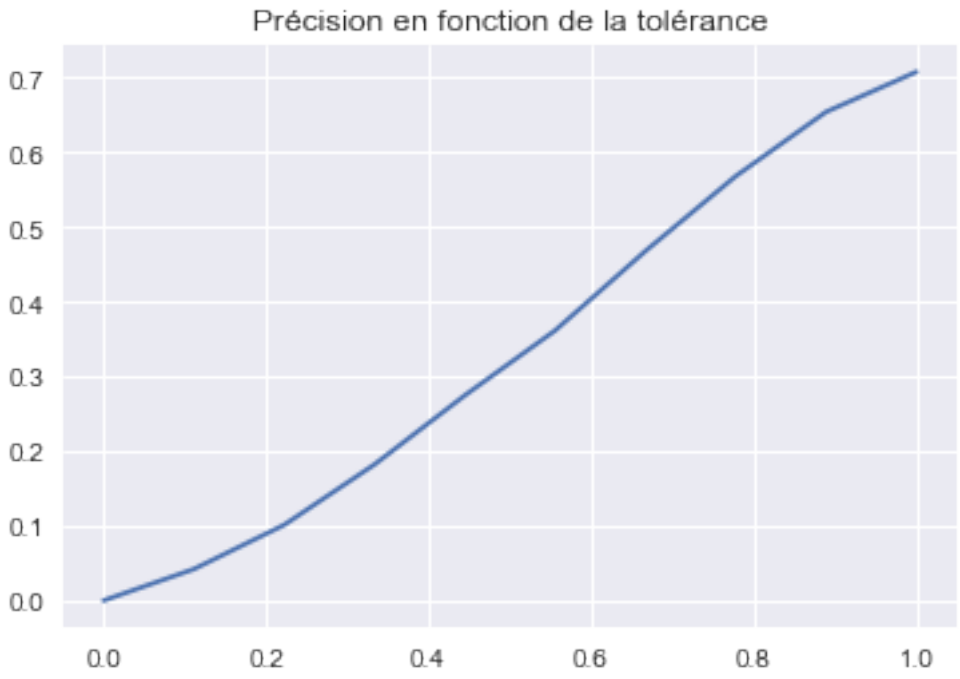
SVR 1 MIN

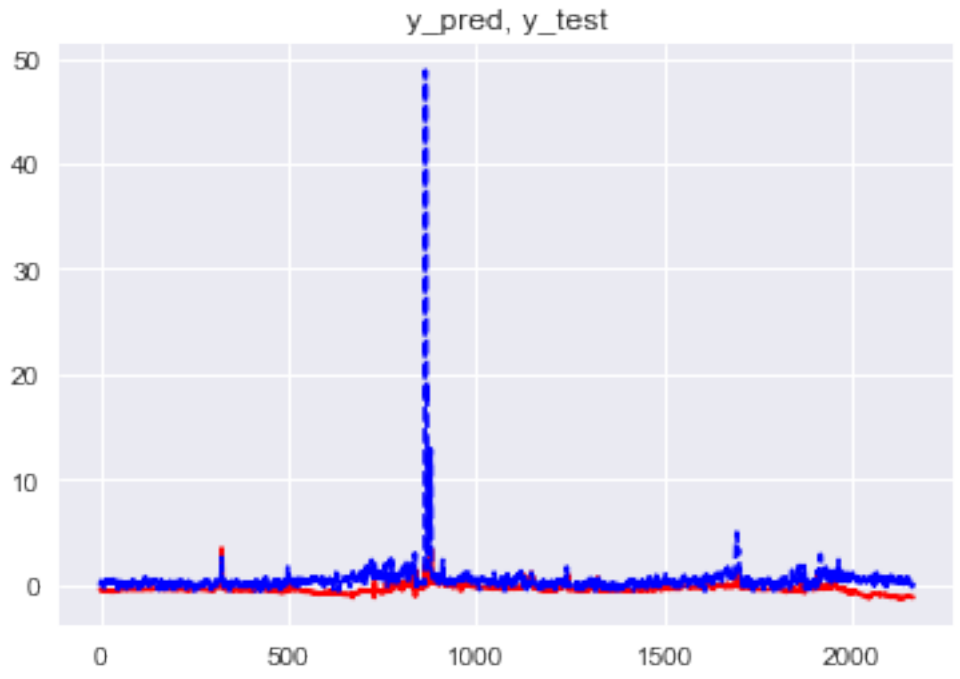
In [14]:
```
ax = sns.heatmap(-scores, cmap="YlGnBu")
```



In [15]:
```
C_idx, gamma_idx = np.unravel_index(scores.argmax(), scores.shape) # Meilleur résulta
clf = svm.SVR(C=C_grid[C_idx], gamma=gamma_grid[gamma_idx])
clf.fit(X_train_1, y_train_1)
err_test = clf.score(X_test_1, y_test_1)
```

In [16]:
```
y_pred = clf.predict(X_test_1)
tolerances = np.linspace(0, 1, 10)
acc = []
for tol in tolerances:
    acc.append(np.sum(np.abs(y_pred.flatten() - y_test_1.flatten()) <= tol) / len(y_te

plt.figure()
plt.title('Précision en fonction de la tolérance')
plt.plot(tolerances, acc)
plt.show()

plt.figure()
plt.title('y_pred, y_test')
plt.plot(y_pred, '-r')
plt.plot(y_test_1, '--b')
plt.show()
```


Précision en fonction de la tolérance
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0.0
0.2
0.4
0.6
0.8
1.0


y_pred, y_test
50
40
30
20
10
0
0
500
1000
1500
2000

2.3 Réseaux de neurones

```
In [17]: from keras.models import Sequential
         from keras.layers import Dense
         from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

```
In [18]: def baseline_model(dense_size, input_dim, loss_function, optimizer):
             # create model
             model = Sequential()
             model.add(Dense(dense_size, input_dim=input_dim, kernel_initializer='normal', acti
             model.add(Dense(1, kernel_initializer='normal'))
             # Compile model
             model.compile(loss=loss_function, optimizer=optimizer)
             model.summary()
             return model

         p = 10
         input_dim = 5
         models_info_15 = {
             'models': {},
             'loss_function': 'mse',
             'optimizer': 'adam',
             'name': '1 Layer: {} neurons'
         }
         #early_stopping = EarlyStopping(monitor='val_loss', verbose=1, mode='auto', patience=
         tolerances = np.linspace(0, 1, 10)

         for i in range(p):
             info_dict = {}

             model = baseline_model(2**(i+1), input_dim, 'mse', 'adagrad')
             history = model.fit(X_train_15, y_train_15, batch_size=5, epochs=100, validation_d
             info_dict['loss'] = history.history['loss']
             info_dict['val_loss'] = history.history['val_loss']

             info_dict['score'] = model.evaluate(X_test_15, y_test_15, batch_size=5)
             y_pred = model.predict(X_test_15)
             acc = []
             for tol in tolerances:
                 y_tol = tol*y_train_15.flatten()
                 accur = np.sum(np.abs(y_pred.flatten() - y_test_15.flatten()) <= tol) / len(y_
                 acc.append(accur)
             info_dict['accuracies'] = acc
             models_info_15['models'][2**(i+1)] = info_dict
```

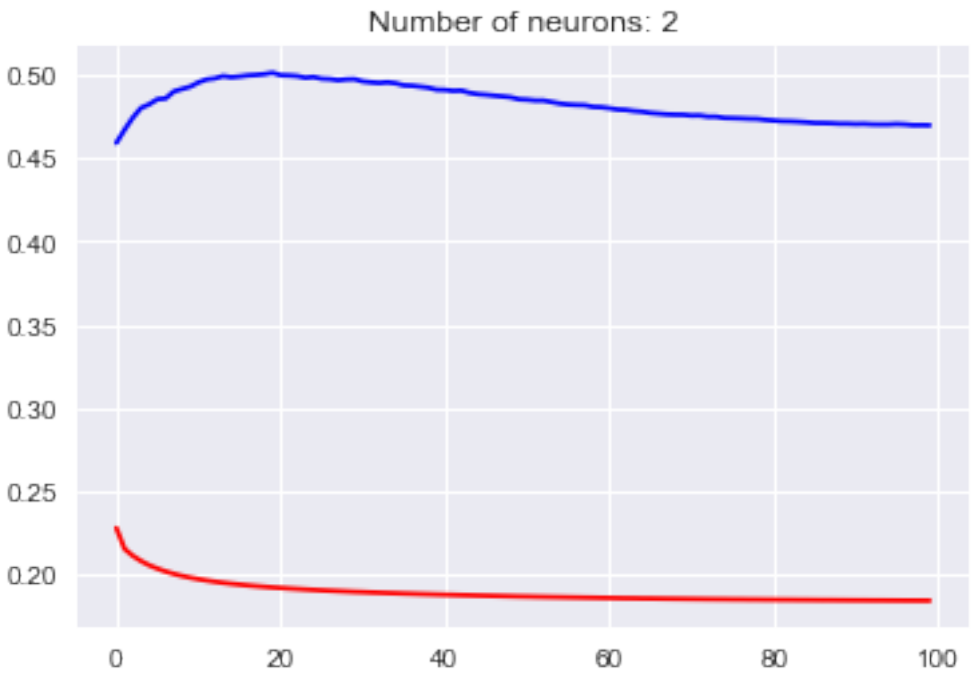
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
```

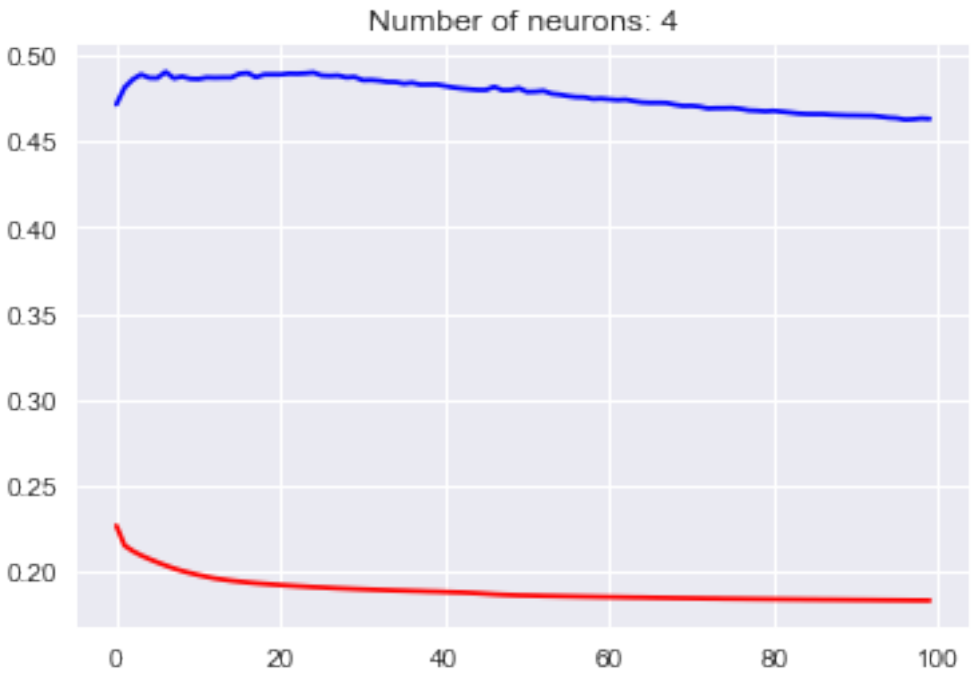
```
=================================================================
dense_1 (Dense)              (None, 2)                 12
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 3
=================================================================
Total params: 15
Trainable params: 15
Non-trainable params: 0
_________________________________________________________________
2090/2169 [============================>..] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 4)                 24
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 5
=================================================================
Total params: 29
Trainable params: 29
Non-trainable params: 0
_________________________________________________________________
2115/2169 [============================>.] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 8)                 48
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 9
=================================================================
Total params: 57
Trainable params: 57
Non-trainable params: 0
_________________________________________________________________
2080/2169 [============================>..] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 16)                96
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 17
=================================================================
Total params: 113
Trainable params: 113
Non-trainable params: 0
_________________________________________________________________
1860/2169 [========================>...] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 32)                192
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 33
```

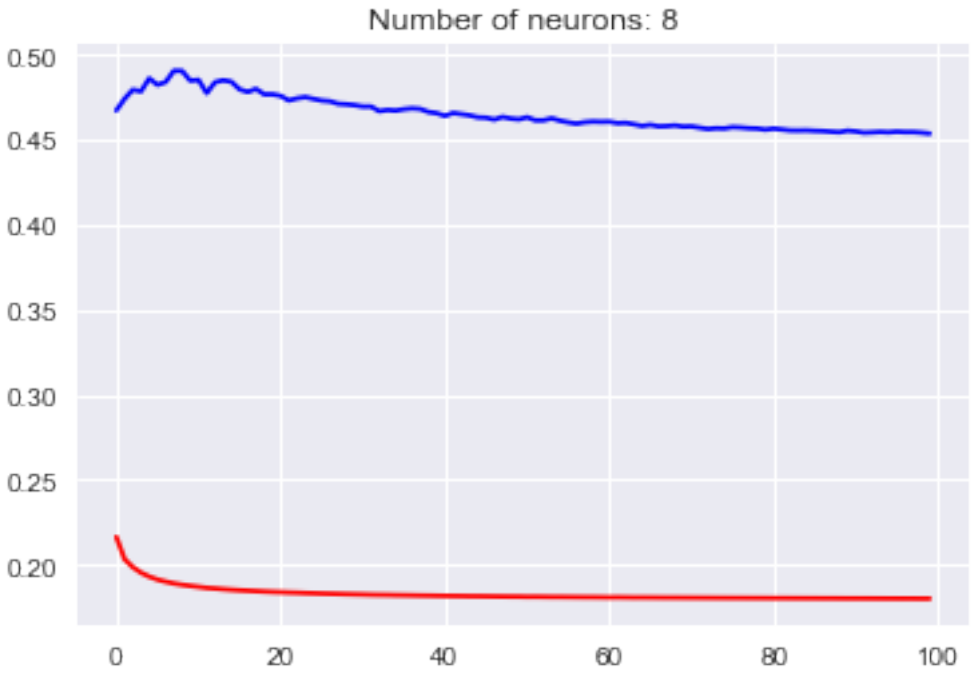
```
=================================================================
Total params: 225
Trainable params: 225
Non-trainable params: 0
_________________________________________________________________
1880/2169 [=========================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_11 (Dense)             (None, 64)                384
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 65
=================================================================
Total params: 449
Trainable params: 449
Non-trainable params: 0
_________________________________________________________________
1965/2169 [==========================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 128)               768
_________________________________________________________________
dense_14 (Dense)             (None, 1)                 129
=================================================================
Total params: 897
Trainable params: 897
Non-trainable params: 0
_________________________________________________________________
2090/2169 [==========================>..] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_15 (Dense)             (None, 256)               1536
_________________________________________________________________
dense_16 (Dense)             (None, 1)                 257
=================================================================
Total params: 1,793
Trainable params: 1,793
Non-trainable params: 0
_________________________________________________________________
1900/2169 [=========================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_17 (Dense)             (None, 512)               3072
_________________________________________________________________
dense_18 (Dense)             (None, 1)                 513
=================================================================
Total params: 3,585
Trainable params: 3,585
Non-trainable params: 0
```

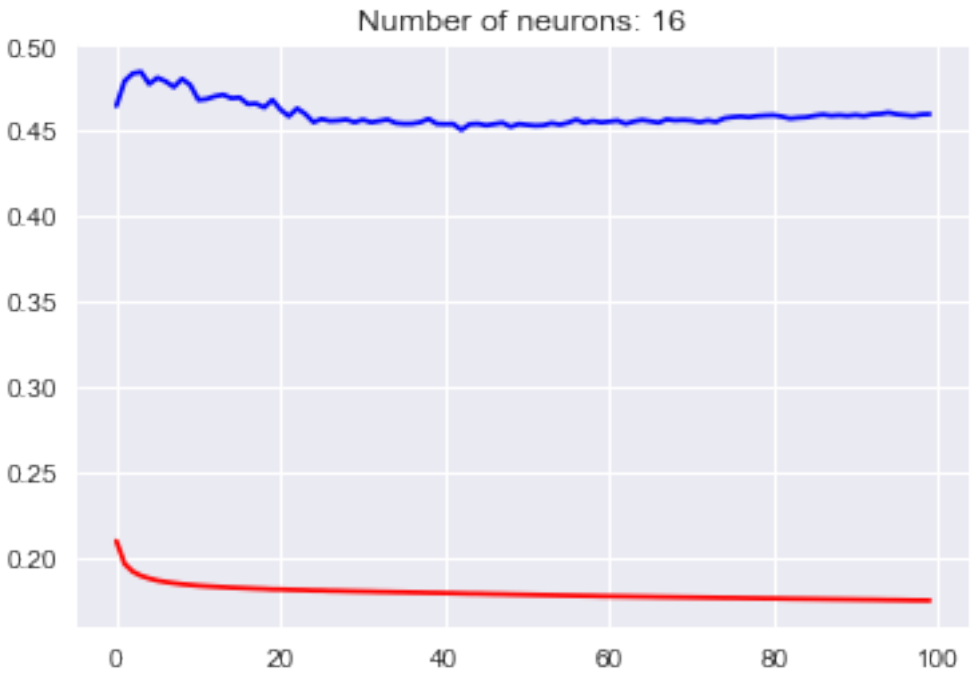
```
_________________________________________________________________
2120/2169 [============================>.] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_19 (Dense)             (None, 1024)              6144
_________________________________________________________________
dense_20 (Dense)             (None, 1)                 1025
=================================================================
Total params: 7,169
Trainable params: 7,169
Non-trainable params: 0
_________________________________________________________________
2169/2169 [==============================] - 0s
```

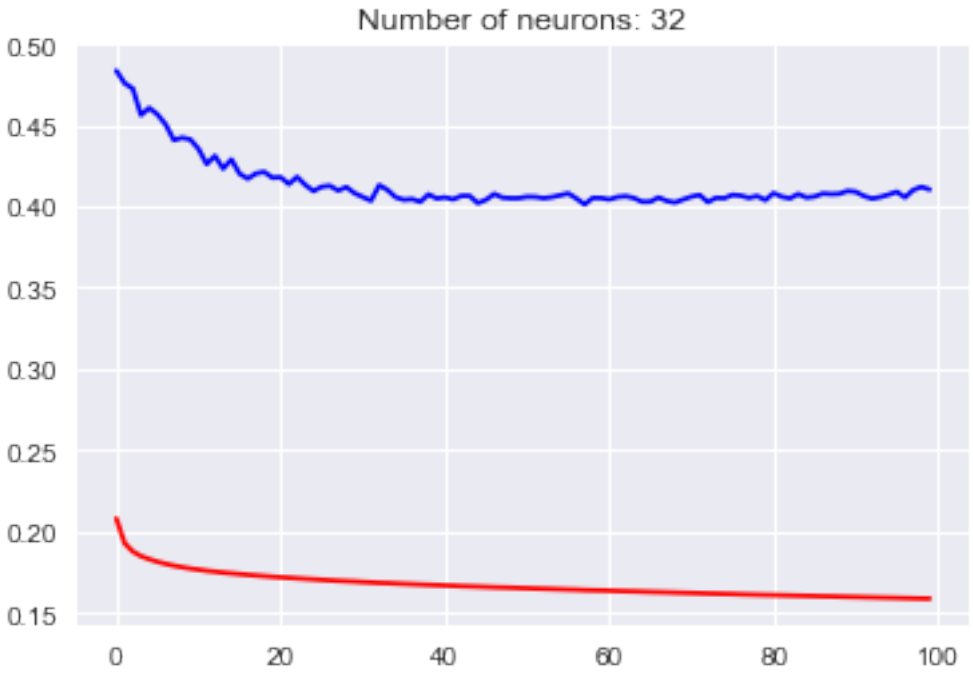
```
In [19]: nb_models = len(models_info_15)
         for nb_neurons, info_dict in models_info_15['models'].items():
             plt.plot(info_dict['loss'], 'r-')
             plt.plot(info_dict['val_loss'], 'b-')
             plt.title('Number of neurons: {}'.format(str(nb_neurons)))
             plt.show()
```

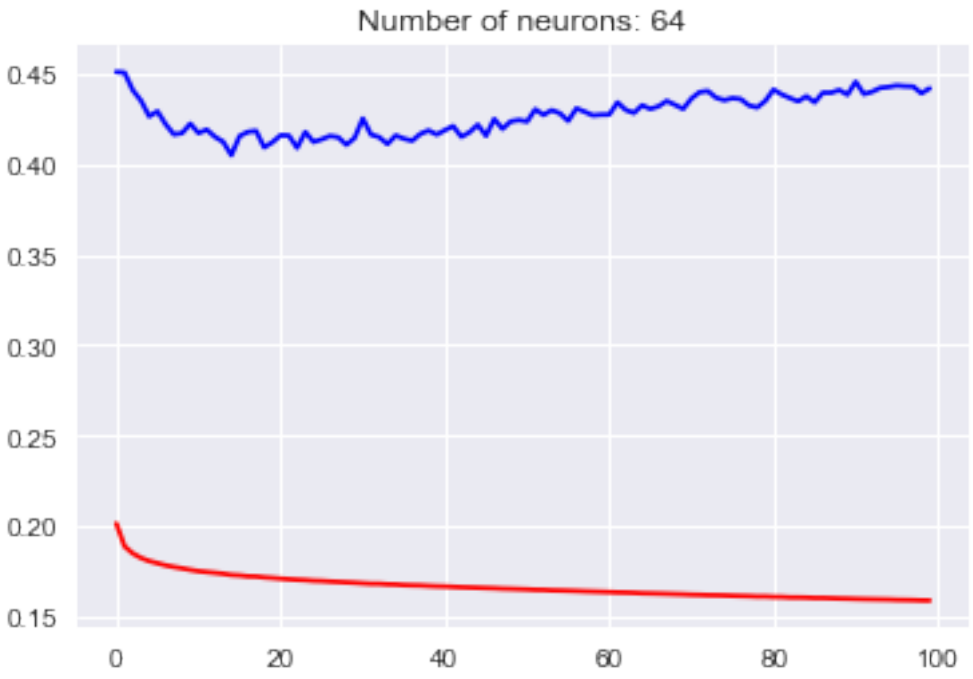


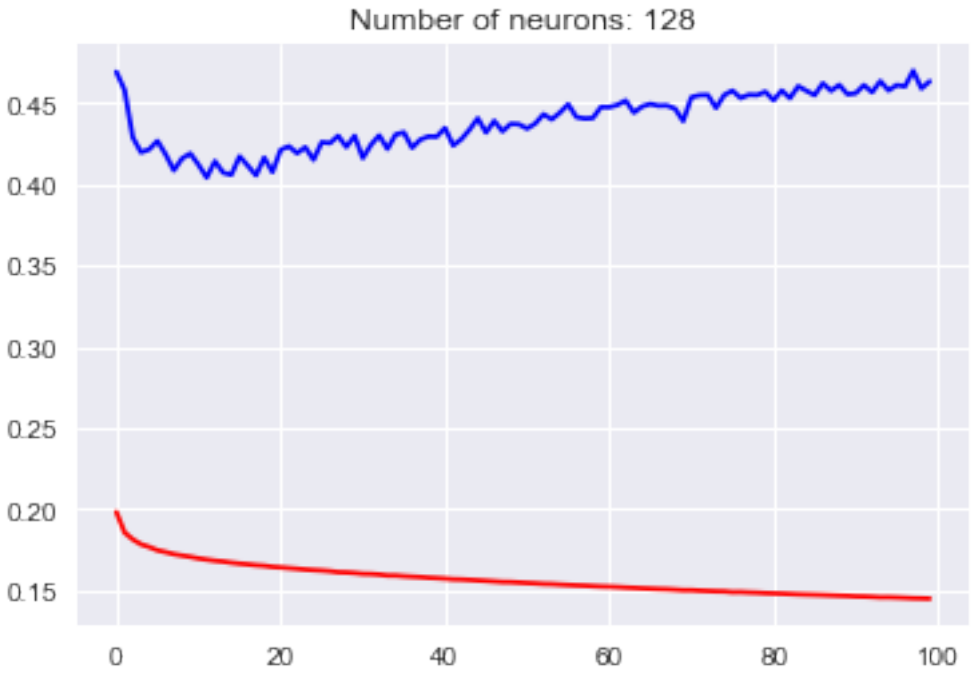

Number of neurons: 4
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
20
40
60
80
100

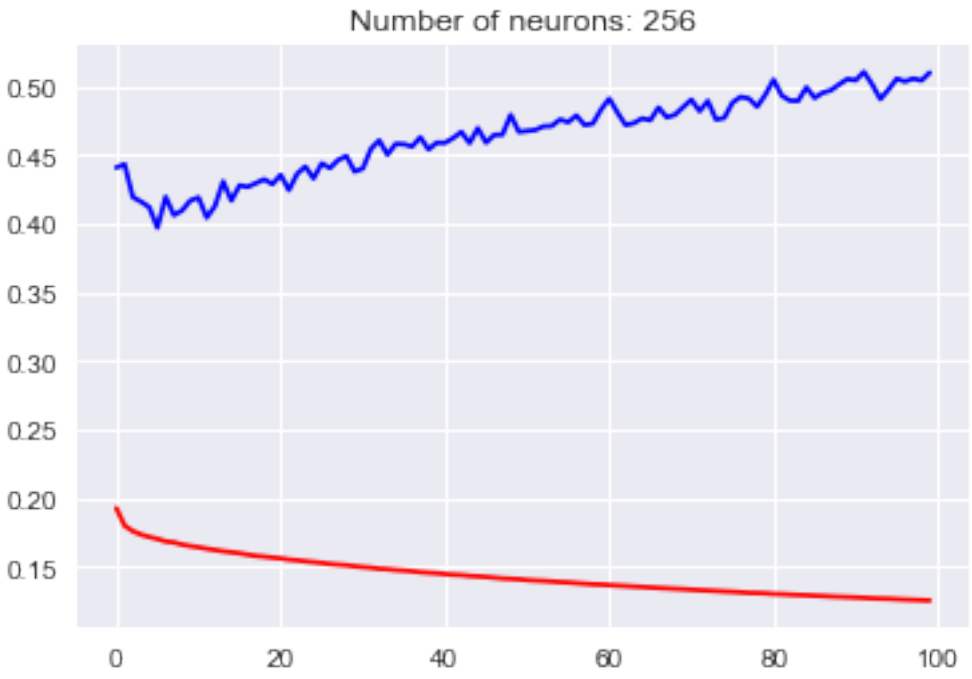

Number of neurons: 8
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
20
40
60
80
100

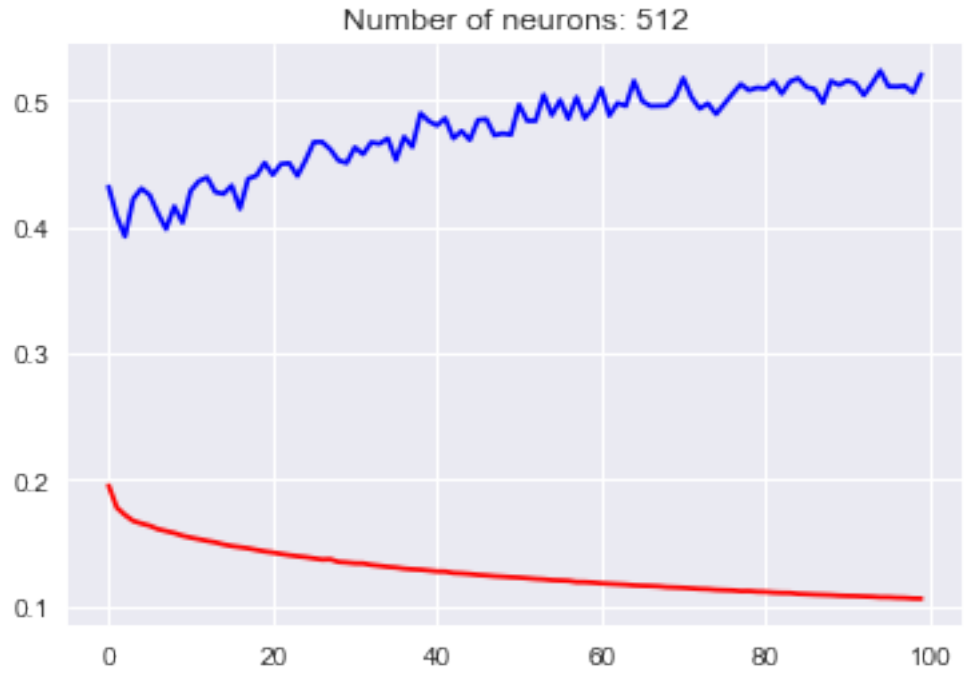

Number of neurons: 16
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0
20
40
60
80
100

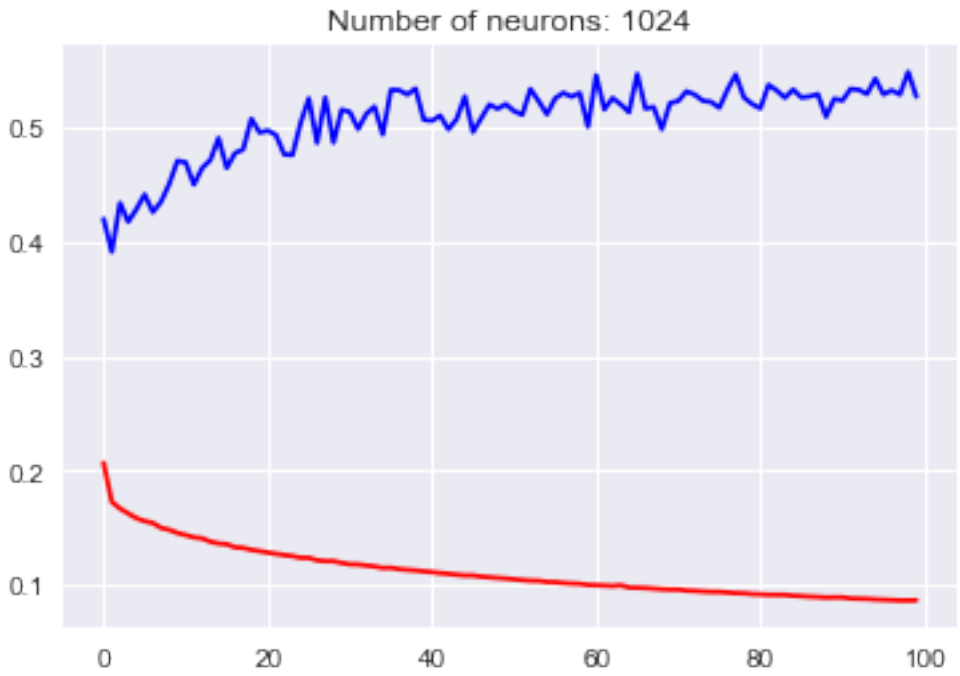

Number of neurons: 32
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 64
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 128
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 256
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 512
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100



In [20]:
```
p = 10
input_dim = 19
models_info_1 = {
    'models': {},
    'loss_function': 'mse',
    'optimizer': 'adagrad',
    'name': '1 Layer: {} neurons'
}

tolerances = np.linspace(0, 1, 10)
for i in range(p):
    info_dict = {}
    model = baseline_model(2**(i+1), input_dim, 'mse', 'adagrad')
    history = model.fit(X_train_1, y_train_1, batch_size=5, epochs=100, validation_da
    info_dict['loss'] = history.history['loss']
    info_dict['val_loss'] = history.history['val_loss']

    info_dict['score'] = model.evaluate(X_test_1, y_test_1, batch_size=5)
    y_pred = model.predict(X_test_1)
    acc = []
    for tol in tolerances:
        y_tol = tol*y_train_1.flatten()
        accur = np.sum(np.abs(y_pred.flatten() - y_test_1.flatten()) <= tol) / len(y_
        acc.append(accur)
```

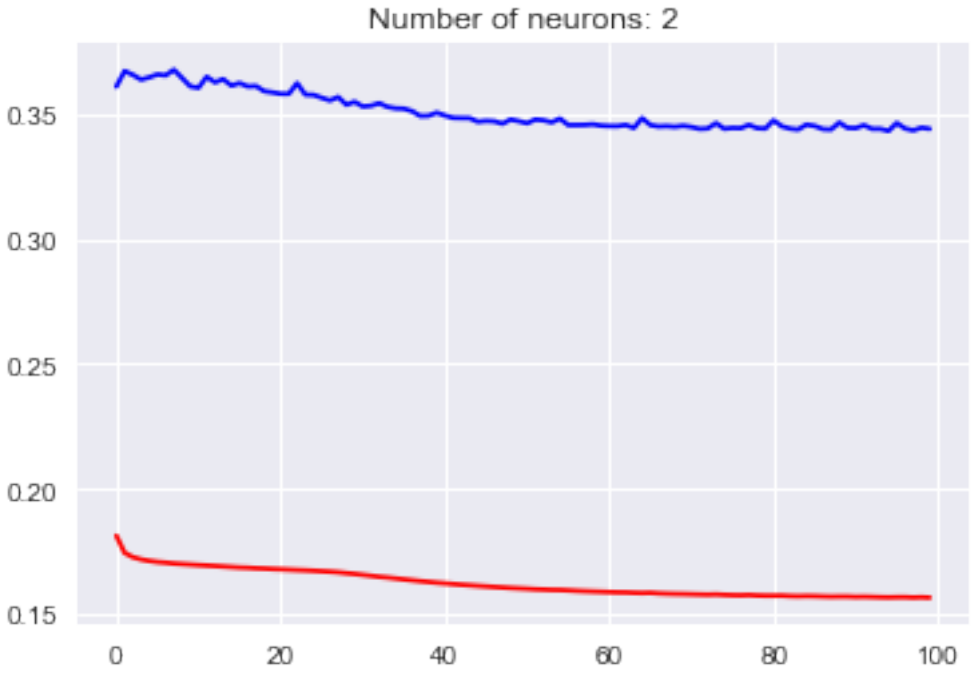
```
            info_dict['accuracies'] = acc
            models_info_1['models'][2**(i+1)] = info_dict
```

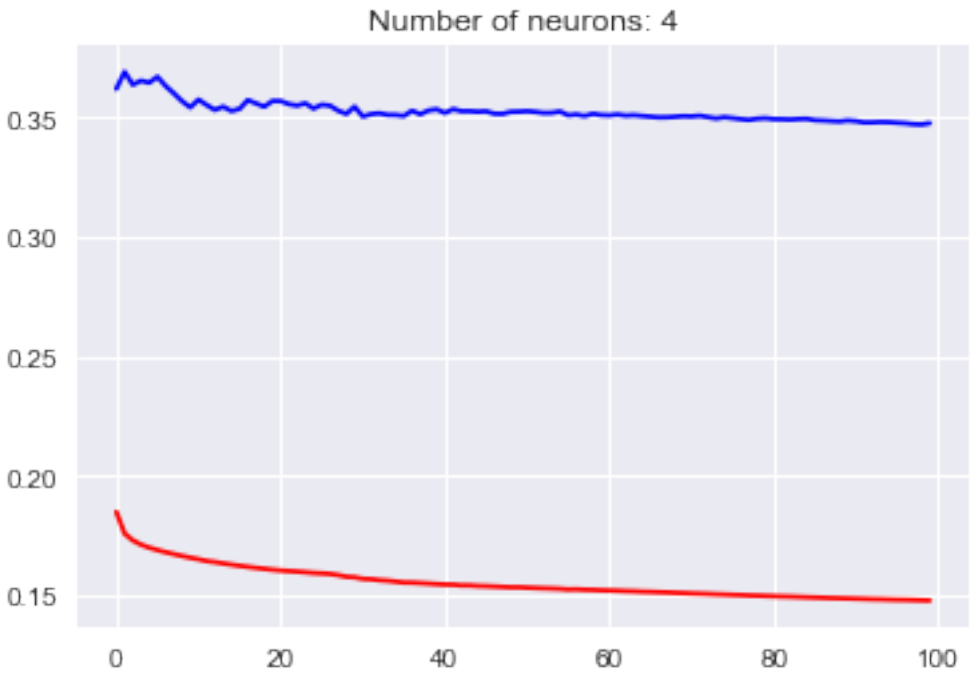
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_21 (Dense)             (None, 2)                 40
_________________________________________________________________
dense_22 (Dense)             (None, 1)                 3
=================================================================
Total params: 43
Trainable params: 43
Non-trainable params: 0
_________________________________________________________________
2140/2163 [============================>.] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_23 (Dense)             (None, 4)                 80
_________________________________________________________________
dense_24 (Dense)             (None, 1)                 5
=================================================================
Total params: 85
Trainable params: 85
Non-trainable params: 0
_________________________________________________________________
1935/2163 [=========================>...] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_25 (Dense)             (None, 8)                 160
_________________________________________________________________
dense_26 (Dense)             (None, 1)                 9
=================================================================
Total params: 169
Trainable params: 169
Non-trainable params: 0
_________________________________________________________________
1900/2163 [=========================>...] - ETA: 0s_____________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 16)                320
_________________________________________________________________
dense_28 (Dense)             (None, 1)                 17
=================================================================
Total params: 337
Trainable params: 337
Non-trainable params: 0
_________________________________________________________________
2080/2163 [============================>..] - ETA: 0s_____________________________________
```

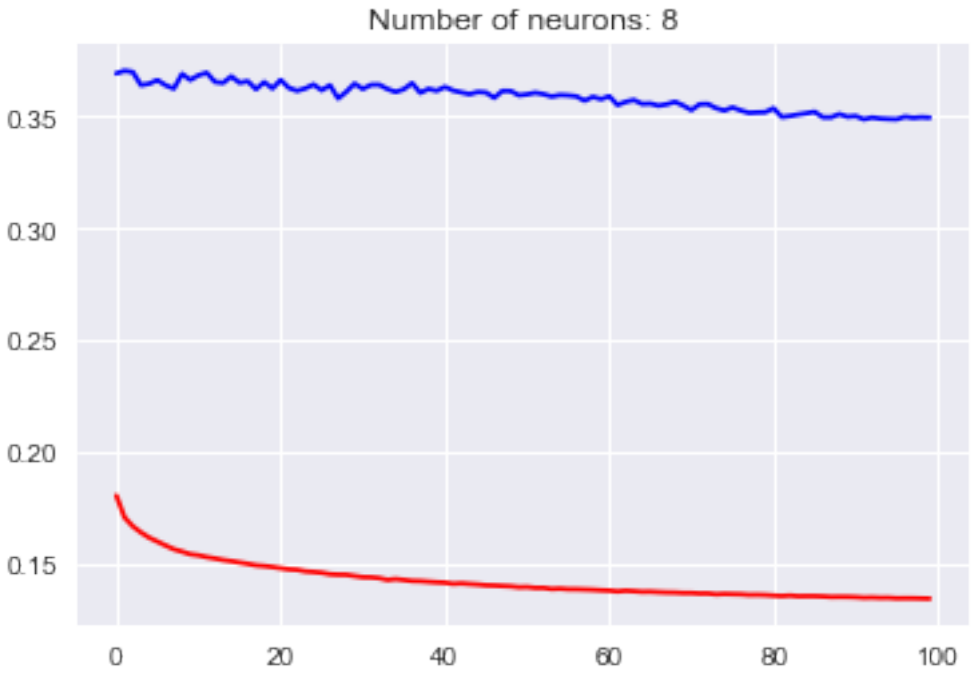
```
Layer (type)                 Output Shape              Param #
=================================================================
dense_29 (Dense)             (None, 32)                640
_________________________________________________________________
dense_30 (Dense)             (None, 1)                 33
=================================================================
Total params: 673
Trainable params: 673
Non-trainable params: 0
_________________________________________________________________
1910/2163 [=======================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_31 (Dense)             (None, 64)                1280
_________________________________________________________________
dense_32 (Dense)             (None, 1)                 65
=================================================================
Total params: 1,345
Trainable params: 1,345
Non-trainable params: 0
_________________________________________________________________
1995/2163 [=========================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_33 (Dense)             (None, 128)               2560
_________________________________________________________________
dense_34 (Dense)             (None, 1)                 129
=================================================================
Total params: 2,689
Trainable params: 2,689
Non-trainable params: 0
_________________________________________________________________
1870/2163 [=======================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_35 (Dense)             (None, 256)               5120
_________________________________________________________________
dense_36 (Dense)             (None, 1)                 257
=================================================================
Total params: 5,377
Trainable params: 5,377
Non-trainable params: 0
_________________________________________________________________
1850/2163 [=======================>...] - ETA: 0s_____________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_37 (Dense)             (None, 512)               10240
_________________________________________________________________
```

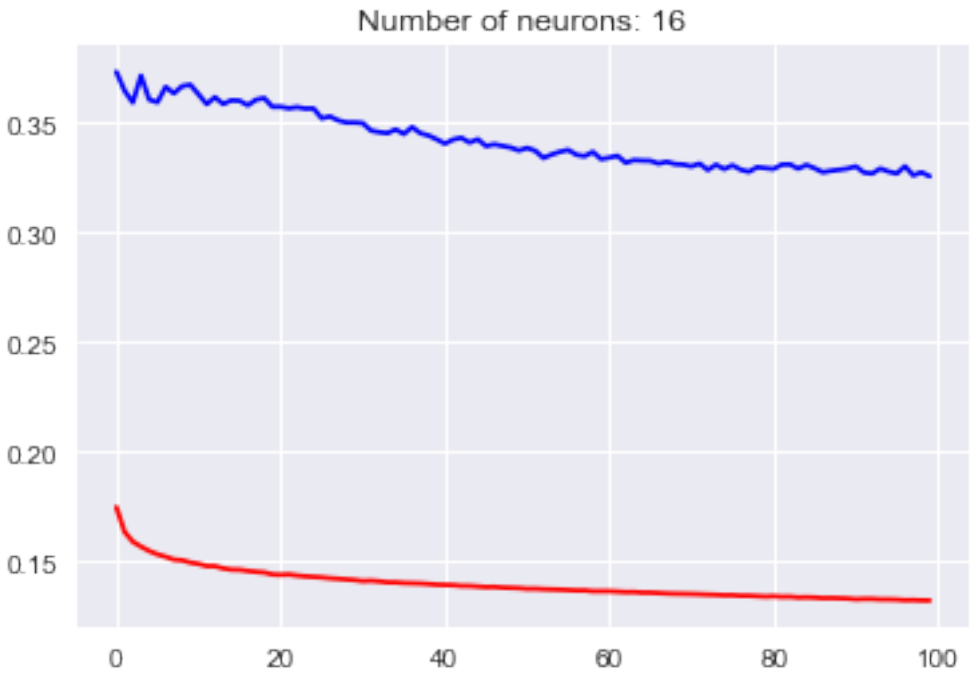
```
dense_38 (Dense)             (None, 1)                 513
=================================================================
Total params: 10,753
Trainable params: 10,753
Non-trainable params: 0
_________________________________________________________________
1845/2163 [========================>...] - ETA: 0s_________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_39 (Dense)             (None, 1024)              20480
_________________________________________________________________
dense_40 (Dense)             (None, 1)                 1025
=================================================================
Total params: 21,505
Trainable params: 21,505
Non-trainable params: 0
_________________________________________________________________
2120/2163 [============================>.] - ETA: 0s
```

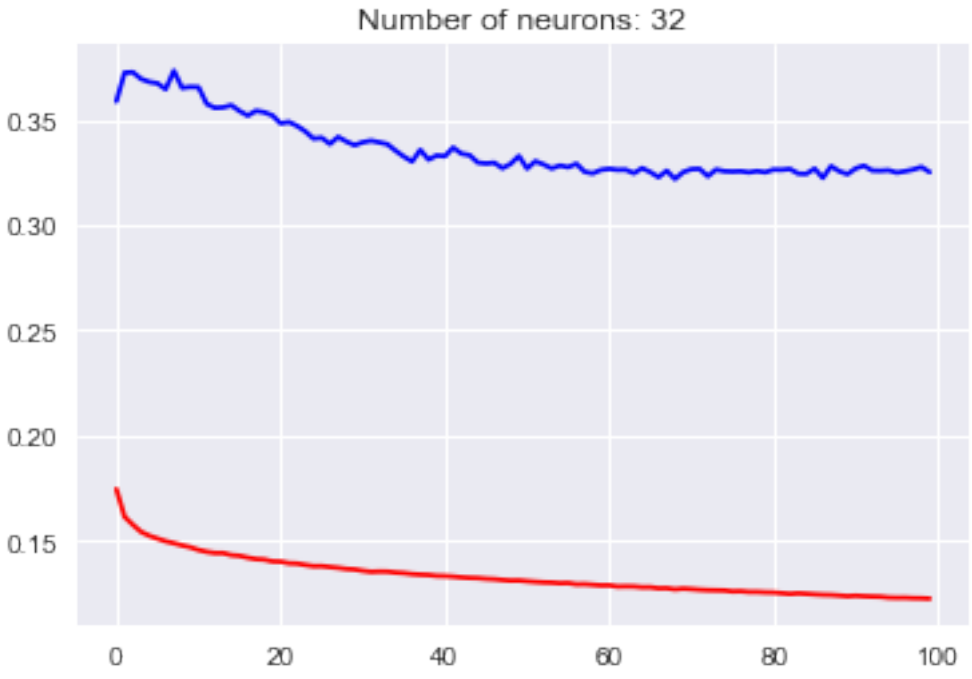
```
In [21]: nb_models = len(models_info_1)
         for nb_neurons, info_dict in models_info_1['models'].items():
             plt.plot(info_dict['loss'], 'r-')
             plt.plot(info_dict['val_loss'], 'b-')
             plt.title('Number of neurons: {}'.format(str(nb_neurons)))
             plt.show()
```

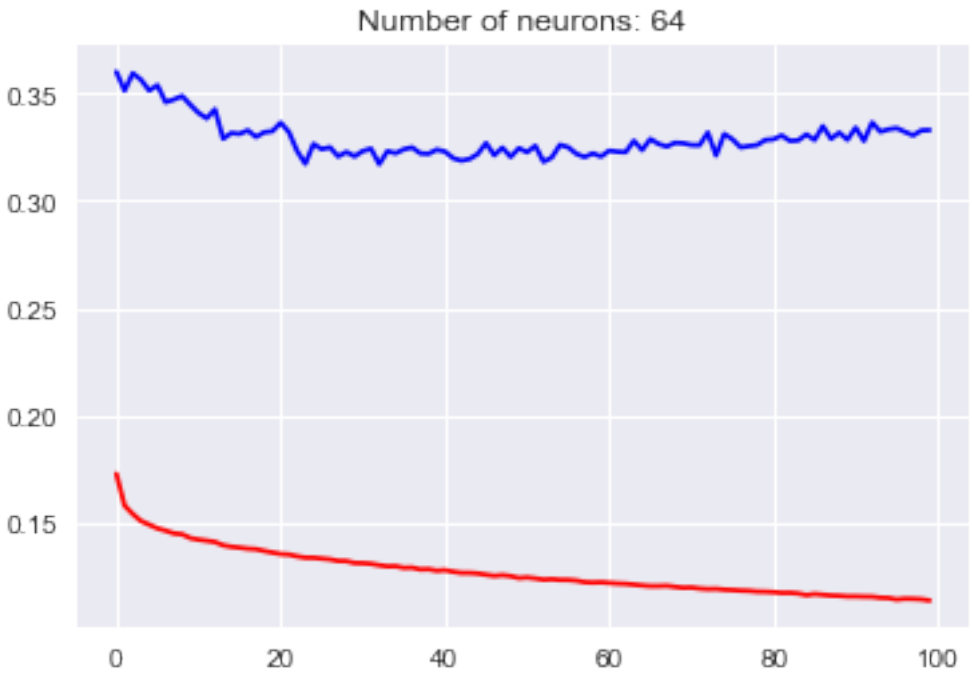


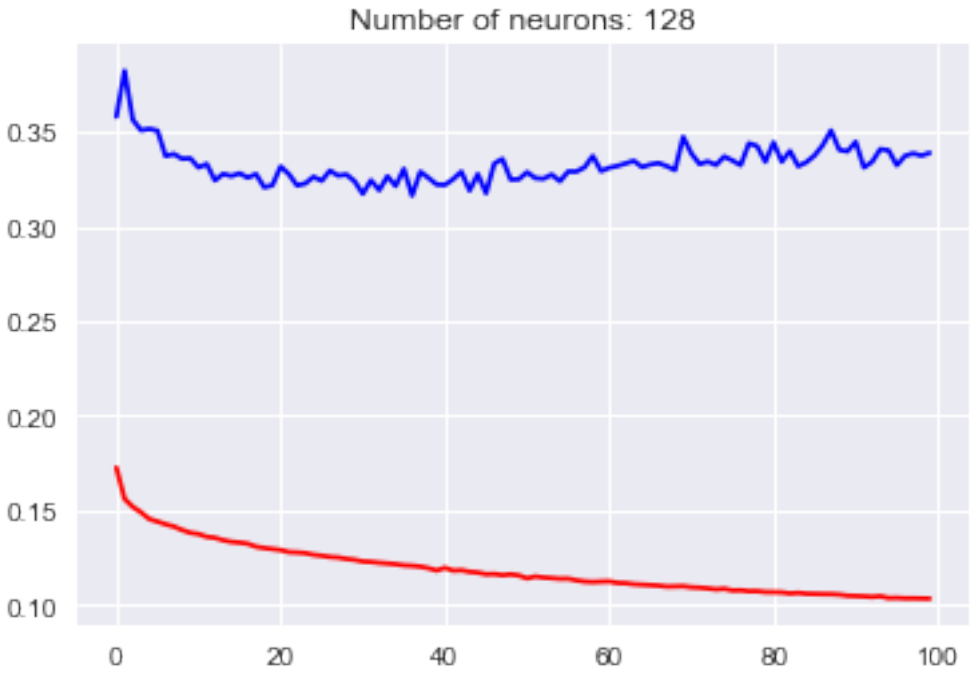

Number of neurons: 4
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100

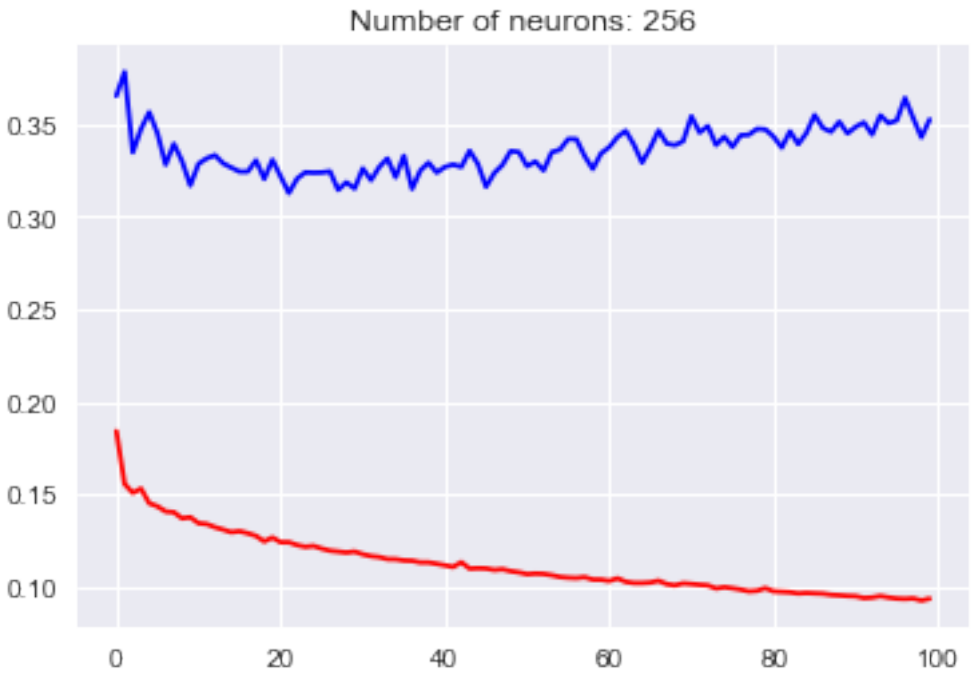

Number of neurons: 8
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100

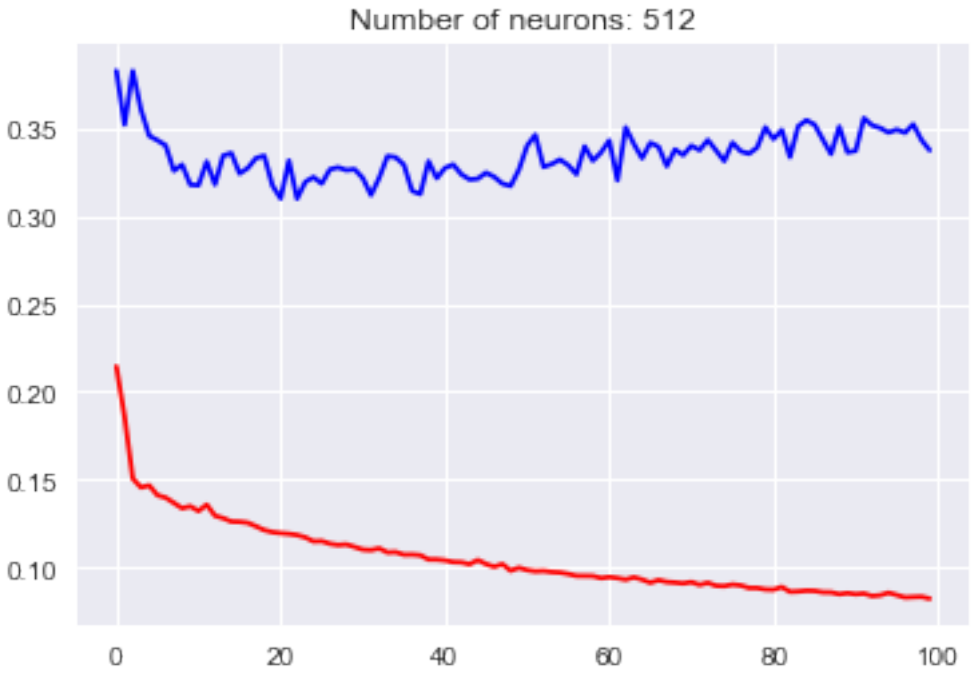

Number of neurons: 16
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100

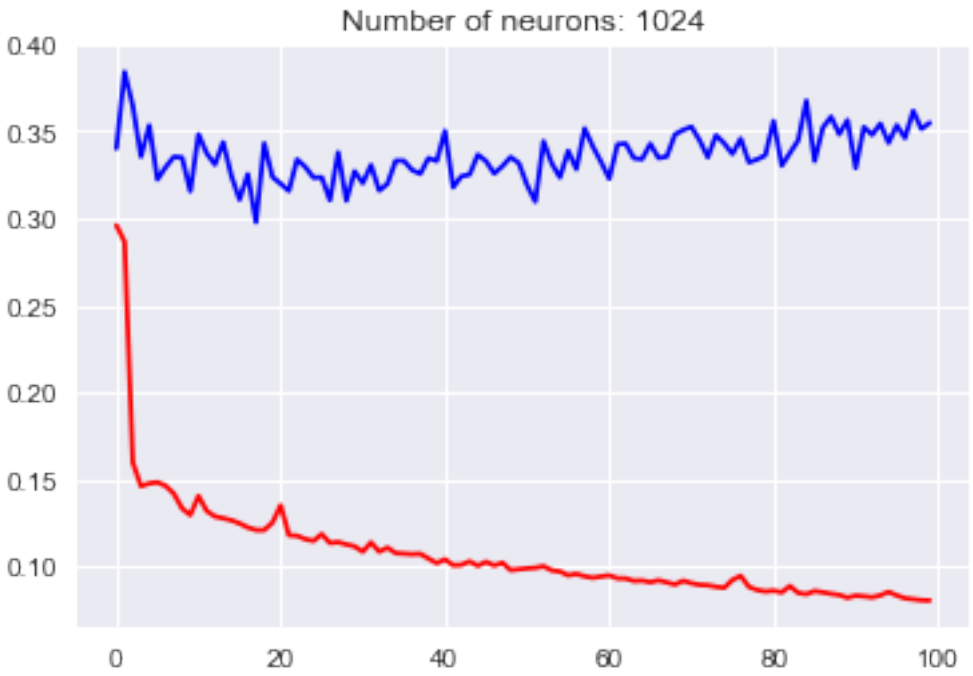

Number of neurons: 32
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 64
0.35
0.30
0.25
0.20
0.15
0
20
40
60
80
100


Number of neurons: 128
0.35
0.30
0.25
0.20
0.15
0.10
0
20
40
60
80
100


Number of neurons: 256
0.35
0.30
0.25
0.20
0.15
0.10
0
20
40
60
80
100


Number of neurons: 512
0.35
0.30
0.25
0.20
0.15
0.10
0
20
40
60
80
100



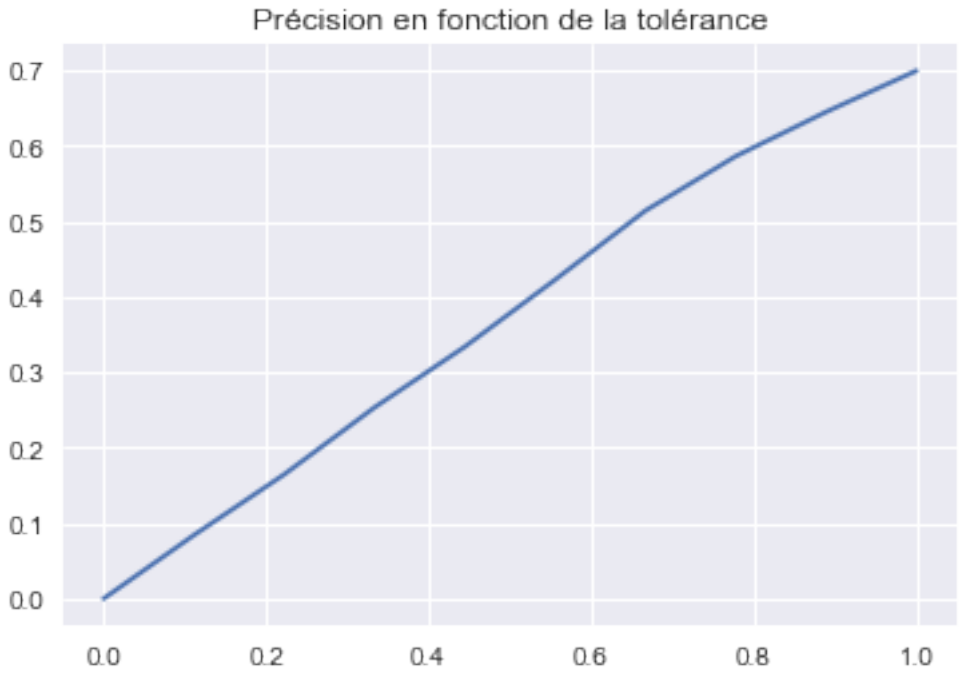

```
In [22]: model = baseline_model(512, input_dim, 'mse', 'adagrad')
         early_stopping = EarlyStopping(monitor='val_loss', verbose=1, mode='auto', patience=1
         history = model.fit(X_train_1, y_train_1, batch_size=5, epochs=100, validation_data=(
```

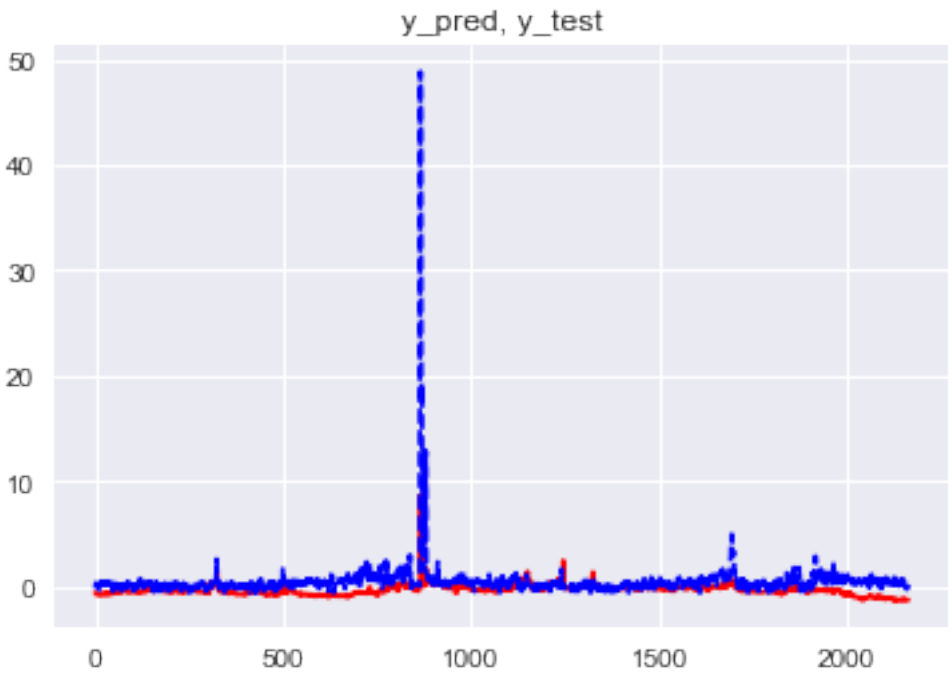
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_41 (Dense)             (None, 512)               10240
_________________________________________________________________
dense_42 (Dense)             (None, 1)                 513
=================================================================
Total params: 10,753
Trainable params: 10,753
Non-trainable params: 0
_________________________________________________________________
Epoch 00028: early stopping
```

```
In [23]: y_pred = model.predict(X_test_1)
         tolerances = np.linspace(0, 1, 10)
         acc = []
         for tol in tolerances:
             acc.append(np.sum(np.abs(y_pred.flatten() - y_test_1.flatten()) <= tol) / len(y_te
```

```
In [24]: plt.figure()
         plt.title('Précision en fonction de la tolérance')
```

```
plt.plot(tolerances, acc)
plt.show()

plt.figure()
plt.title('y_pred, y_test')
plt.plot(y_pred, '-r')
plt.plot(y_test_1, '--b')
plt.show()
```





In [ ]: