Musk Classification

About

The given dataset contains details about organic chemical compounds including their chemical features, isomeric conformation, names and the classes in which they are classified. The compounds are classified as either 'Musk' or 'Non-Musk' compounds.

Your task is to build a classification model on the given data using any Deep Learning approach

Attributes in musk_csv.csv

1. molecule_name: Symbolic name of each molecule.
2. conformation_name: Symbolic name of each conformation
3. f1-f166: These are chemical features.
4. class: 0 if compound is classified as non-musk, 1 if compound is classified as musk

In [1]:

```
import pandas as pd
import numpy as np
from sklearn import preprocessing
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns

from sklearn import preprocessing
from keras.models import Sequential
from keras.layers import Dense,Activation, Dropout
from imblearn.over_sampling import SMOTE

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
```

```
Using TensorFlow backend.
```

Exploratory Data Analysis

In [2]:

```
df=pd.read_csv("musk_csv.csv")
df.head()
```

Out[2]:

	ID	molecule_name	conformation_name	f1	f2	f3	f4	f5	f6	f7	...	f158	f159	f160	f161	f162	f163	f164	f165	f166	cla
0	1	MUSK-211	211_1+1	46	-108	-60	-69	-117	49	38	...	-308	52	-7	39	126	156	-50	-112	96	
1	2	MUSK-211	211_1+10	41	-188	-145	22	-117	-6	57	...	-59	-2	52	103	136	169	-61	-136	79	
2	3	MUSK-211	211_1+11	46	-194	-145	28	-117	73	57	...	-134	-154	57	143	142	165	-67	-145	39	
3	4	MUSK-211	211_1+12	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	136	168	-60	-135	80	
4	5	MUSK-211	211_1+13	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	137	168	-60	-135	80	

5 rows × 170 columns

In [3]:

```
df.tail()
```

Out[3]:

	ID	molecule_name	conformation_name	f1	f2	f3	f4	f5	f6	f7	...	f158	f159	f160	f161	f162	f163	f164	f165
6593	6594	NON-MUSK-jp13	jp13_2+5	51	-123	-23	-108	-117	134	-160	...	-66	164	-14	-29	107	171	-44	-115
6594	6595	NON-MUSK-jp13	jp13_2+6	44	-104	-19	-105	-117	142	-165	...	-51	166	-9	150	129	158	-66	-144
6595	6596	NON-MUSK-jp13	jp13_2+7	44	-102	-19	-104	-117	72	-165	...	90	117	-8	150	130	159	-66	-144
6596	6597	NON-MUSK-jp13	jp13_2+8	51	-121	-23	-106	-117	63	-161	...	86	99	-14	-31	106	171	-44	-116
6597	6598	NON-MUSK-jp13	jp13_2+9	51	-122	-23	-106	-117	190	-161	...	40	124	-14	-30	107	171	-44	-115

5 rows × 170 columns

In [4]:

```
df.shape
```

Out[4]:

```
(6598, 170)
```

In [5]:

```
df.columns
```

Out[5]:

```
Index(['ID', 'molecule_name', 'conformation_name', 'f1', 'f2', 'f3', 'f4',
       'f5', 'f6', 'f7',
       ...
       'f158', 'f159', 'f160', 'f161', 'f162', 'f163', 'f164', 'f165', 'f166',
       'class'],
      dtype='object', length=170)
```

Checking for any null values

In [6]:

```
#Check for null values
df.isnull().any()
```

Out[6]:

```
ID                   False
molecule_name        False
conformation_name    False
f1                   False
f2                   False
f3                   False
f4                   False
f5                   False
f6                   False
f7                   False
f8                   False
f9                   False
f10                  False
f11                  False
f12                  False
```

```
f13                     False
f14                     False
f15                     False
f16                     False
f17                     False
f18                     False
f19                     False
f20                     False
f21                     False
f22                     False
f23                     False
f24                     False
f25                     False
f26                     False
f27                     False
                        ...
f138                    False
f139                    False
f140                    False
f141                    False
f142                    False
f143                    False
f144                    False
f145                    False
f146                    False
f147                    False
f148                    False
f149                    False
f150                    False
f151                    False
f152                    False
f153                    False
f154                    False
f155                    False
f156                    False
f157                    False
f158                    False
f159                    False
f160                    False
f161                    False
f162                    False
f163                    False
f164                    False
f165                    False
f166                    False
class                   False
Length: 170, dtype: bool
```

In [7]:

```
df['class'].value_counts()
```

Out[7]:

```
0    5581
1    1017
Name: class, dtype: int64
```

In [8]:

```
#plotting pie chart for class
df['class'].value_counts().plot(kind='pie', figsize=(5,5))
```

Out[8]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x21ba285be08>
```

Observation : Class variable is imbalanced, will use SMOTE to balance the data

In [9]:

```
# Dropping ID, molecule_name, Conformation_name features from dataframe
df = df.drop(['ID'],axis=1)
df = df.drop(['molecule_name'],axis=1)
df = df.drop(['conformation_name'],axis=1)
```

In [10]:

```
df.columns
```

Out[10]:

```
Index(['f1', 'f2', 'f3', 'f4', 'f5', 'f6', 'f7', 'f8', 'f9', 'f10',
       ...
       'f158', 'f159', 'f160', 'f161', 'f162', 'f163', 'f164', 'f165', 'f166',
       'class'],
      dtype='object', length=167)
```

Splitting the data into 80:20 ratio

In [12]:

```
y=df['class']
X=df.drop('class',axis=1)
```

In [13]:

```
#train test split
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [14]:

```
scaler = preprocessing.StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Resampling using SMOTE to balance data

In [15]:

```
sm = SMOTE(random_state = 2)
X_train_res, y_train_res = sm.fit_sample(X_train, y_train.ravel())
```

In [16]:

```
print("After OverSampling, counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, counts of label '0': {}".format(sum(y_train_res == 0)))
```

```
After OverSampling, counts of label '1': 4470
After OverSampling, counts of label '0': 4470
```

Model 1

In [17]:

```
model = Sequential()
model.add(Dense(64,activation='relu', input_shape=(166,)))
model.add(Dense(32,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
```

```
WARNING: Logging before flag parsing goes to stderr.
W1215 14:19:50.698777  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\keras\backend\tensorflow_backend.py:74: The name tf.get_default_graph is deprecated. Plea
se use tf.compat.v1.get_default_graph instead.

W1215 14:19:50.730056  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\keras\backend\tensorflow_backend.py:517: The name tf.placeholder is deprecated. Please us
e tf.compat.v1.placeholder instead.

W1215 14:19:50.745678  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\keras\backend\tensorflow_backend.py:4138: The name tf.random_uniform is deprecated. Pleas
e use tf.random.uniform instead.
```

In [18]:

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 64)                10688
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 33
=================================================================
Total params: 12,801
Trainable params: 12,801
Non-trainable params: 0
_________________________________________________________________
```

In [19]:

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
W1215 14:20:00.316965  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-packages\keras\optimizers.py:790: Th
e name tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

W1215 14:20:00.348249  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\keras\backend\tensorflow_backend.py:3376: The name tf.log is deprecated. Please use tf.ma
th.log instead.

W1215 14:20:00.363829  8876 deprecation.py:323] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_support.<locals>.wrapper (from
tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

In [20]:

```
history = model.fit(X_train_res,y_train_res,epochs=50,batch_size=128,validation_data=(X_test,y_test
```

```
))
score = model.evaluate(X_test,y_test,verbose=0)
```

```
W1215 14:20:48.797669  8876 deprecation_wrapper.py:119] From
c:\users\acer\appdata\local\programs\python\python37\lib\site-
packages\keras\backend\tensorflow_backend.py:986: The name tf.assign_add is deprecated. Please use
tf.compat.v1.assign_add instead.
```

```
Train on 8940 samples, validate on 1320 samples
Epoch 1/50
8940/8940 [==============================] - 1s 79us/step - loss: 0.3586 - acc: 0.8451 - val_loss:
0.1907 - val_acc: 0.9265
Epoch 2/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.1426 - acc: 0.9503 - val_loss:
0.1134 - val_acc: 0.9629
Epoch 3/50
8940/8940 [==============================] - 0s 24us/step - loss: 0.0850 - acc: 0.9729 - val_loss:
0.0891 - val_acc: 0.9705
Epoch 4/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0581 - acc: 0.9834 - val_loss:
0.0603 - val_acc: 0.9773
Epoch 5/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0400 - acc: 0.9890 - val_loss:
0.0502 - val_acc: 0.9856
Epoch 6/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0293 - acc: 0.9927 - val_loss:
0.0403 - val_acc: 0.9856
Epoch 7/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0213 - acc: 0.9957 - val_loss:
0.0357 - val_acc: 0.9871
Epoch 8/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0154 - acc: 0.9970 - val_loss:
0.0298 - val_acc: 0.9894
Epoch 9/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0126 - acc: 0.9981 - val_loss:
0.0253 - val_acc: 0.9917
Epoch 10/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0085 - acc: 0.9990 - val_loss:
0.0269 - val_acc: 0.9894
Epoch 11/50
8940/8940 [==============================] - 0s 21us/step - loss: 0.0067 - acc: 0.9996 - val_loss:
0.0206 - val_acc: 0.9917
Epoch 12/50
8940/8940 [==============================] - 0s 26us/step - loss: 0.0051 - acc: 0.9998 - val_loss:
0.0184 - val_acc: 0.9917
Epoch 13/50
8940/8940 [==============================] - 0s 26us/step - loss: 0.0043 - acc: 0.9997 - val_loss:
0.0164 - val_acc: 0.9939
Epoch 14/50
8940/8940 [==============================] - 0s 24us/step - loss: 0.0029 - acc: 1.0000 - val_loss:
0.0159 - val_acc: 0.9932
Epoch 15/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0023 - acc: 1.0000 - val_loss:
0.0140 - val_acc: 0.9955
Epoch 16/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0019 - acc: 1.0000 - val_loss:
0.0135 - val_acc: 0.9947
Epoch 17/50
8940/8940 [==============================] - 0s 26us/step - loss: 0.0017 - acc: 1.0000 - val_loss:
0.0144 - val_acc: 0.9947
Epoch 18/50
8940/8940 [==============================] - 0s 24us/step - loss: 0.0015 - acc: 1.0000 - val_loss:
0.0145 - val_acc: 0.9939
Epoch 19/50
8940/8940 [==============================] - 0s 24us/step - loss: 0.0012 - acc: 1.0000 - val_loss:
0.0135 - val_acc: 0.9939
Epoch 20/50
8940/8940 [==============================] - 0s 23us/step - loss: 0.0010 - acc: 1.0000 - val_loss:
0.0135 - val_acc: 0.9947
Epoch 21/50
8940/8940 [==============================] - 0s 24us/step - loss: 9.1424e-04 - acc: 1.0000 - val_l
oss: 0.0125 - val_acc: 0.9947
Epoch 22/50
8940/8940 [==============================] - 0s 28us/step - loss: 9.8869e-04 - acc: 1.0000 - val_l
```

```
oss: 0.0142 - val_acc: 0.9947
Epoch 23/50
8940/8940 [==============================] - 0s 26us/step - loss: 7.6004e-04 - acc: 1.0000 - val_l
oss: 0.0113 - val_acc: 0.9962
Epoch 24/50
8940/8940 [==============================] - 0s 28us/step - loss: 6.1254e-04 - acc: 1.0000 - val_l
oss: 0.0113 - val_acc: 0.9962
Epoch 25/50
8940/8940 [==============================] - 0s 24us/step - loss: 5.6481e-04 - acc: 1.0000 - val_l
oss: 0.0112 - val_acc: 0.9955
Epoch 26/50
8940/8940 [==============================] - 0s 26us/step - loss: 4.8307e-04 - acc: 1.0000 - val_l
oss: 0.0113 - val_acc: 0.9939
Epoch 27/50
8940/8940 [==============================] - 0s 26us/step - loss: 4.4662e-04 - acc: 1.0000 - val_l
oss: 0.0110 - val_acc: 0.9947
Epoch 28/50
8940/8940 [==============================] - 0s 28us/step - loss: 4.1043e-04 - acc: 1.0000 - val_l
oss: 0.0117 - val_acc: 0.9955
Epoch 29/50
8940/8940 [==============================] - 0s 24us/step - loss: 3.6552e-04 - acc: 1.0000 - val_l
oss: 0.0106 - val_acc: 0.9955
Epoch 30/50
8940/8940 [==============================] - 0s 26us/step - loss: 3.3483e-04 - acc: 1.0000 - val_l
oss: 0.0106 - val_acc: 0.9955
Epoch 31/50
8940/8940 [==============================] - 0s 26us/step - loss: 3.1827e-04 - acc: 1.0000 - val_l
oss: 0.0111 - val_acc: 0.9955
Epoch 32/50
8940/8940 [==============================] - 0s 24us/step - loss: 3.0255e-04 - acc: 1.0000 - val_l
oss: 0.0098 - val_acc: 0.9962
Epoch 33/50
8940/8940 [==============================] - 0s 24us/step - loss: 2.6181e-04 - acc: 1.0000 - val_l
oss: 0.0105 - val_acc: 0.9970
Epoch 34/50
8940/8940 [==============================] - 0s 26us/step - loss: 2.3607e-04 - acc: 1.0000 - val_l
oss: 0.0108 - val_acc: 0.9955
Epoch 35/50
8940/8940 [==============================] - 0s 28us/step - loss: 2.2556e-04 - acc: 1.0000 - val_l
oss: 0.0105 - val_acc: 0.9970
Epoch 36/50
8940/8940 [==============================] - 0s 26us/step - loss: 2.0181e-04 - acc: 1.0000 - val_l
oss: 0.0104 - val_acc: 0.9970
Epoch 37/50
8940/8940 [==============================] - 0s 26us/step - loss: 1.8711e-04 - acc: 1.0000 - val_l
oss: 0.0099 - val_acc: 0.9962
Epoch 38/50
8940/8940 [==============================] - 0s 24us/step - loss: 1.7457e-04 - acc: 1.0000 - val_l
oss: 0.0103 - val_acc: 0.9970
Epoch 39/50
8940/8940 [==============================] - 0s 26us/step - loss: 1.6303e-04 - acc: 1.0000 - val_l
oss: 0.0097 - val_acc: 0.9970
Epoch 40/50
8940/8940 [==============================] - 0s 26us/step - loss: 1.4977e-04 - acc: 1.0000 - val_l
oss: 0.0094 - val_acc: 0.9977
Epoch 41/50
8940/8940 [==============================] - 0s 28us/step - loss: 1.3900e-04 - acc: 1.0000 - val_l
oss: 0.0096 - val_acc: 0.9977
Epoch 42/50
8940/8940 [==============================] - 0s 28us/step - loss: 1.3054e-04 - acc: 1.0000 - val_l
oss: 0.0102 - val_acc: 0.9970
Epoch 43/50
8940/8940 [==============================] - 0s 24us/step - loss: 1.2652e-04 - acc: 1.0000 - val_l
oss: 0.0091 - val_acc: 0.9977
Epoch 44/50
8940/8940 [==============================] - 0s 24us/step - loss: 1.1603e-04 - acc: 1.0000 - val_l
oss: 0.0096 - val_acc: 0.9970
Epoch 45/50
8940/8940 [==============================] - 0s 28us/step - loss: 1.0964e-04 - acc: 1.0000 - val_l
oss: 0.0098 - val_acc: 0.9970
Epoch 46/50
8940/8940 [==============================] - 0s 26us/step - loss: 1.0248e-04 - acc: 1.0000 - val_l
oss: 0.0096 - val_acc: 0.9970
Epoch 47/50
8940/8940 [==============================] - 0s 26us/step - loss: 9.4372e-05 - acc: 1.0000 - val_l
oss: 0.0096 - val_acc: 0.9970
Epoch 48/50
```

```
8940/8940 [==============================] - 0s 26us/step - loss: 9.1304e-05 - acc: 1.0000 - val_l
oss: 0.0094 - val_acc: 0.9977
Epoch 49/50
8940/8940 [==============================] - 0s 26us/step - loss: 8.3438e-05 - acc: 1.0000 - val_l
oss: 0.0096 - val_acc: 0.9970
Epoch 50/50
8940/8940 [==============================] - 0s 24us/step - loss: 8.0339e-05 - acc: 1.0000 - val_l
oss: 0.0094 - val_acc: 0.9970
```

In [21]:

```
print("Accuracy : %.2f%%" % (score[1]*100))
```

```
Accuracy : 99.70%
```

Saving h5 model

In [50]:

```
# serialize model to JSON
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
# serialize weights to HDF5
model.save_weights("model.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

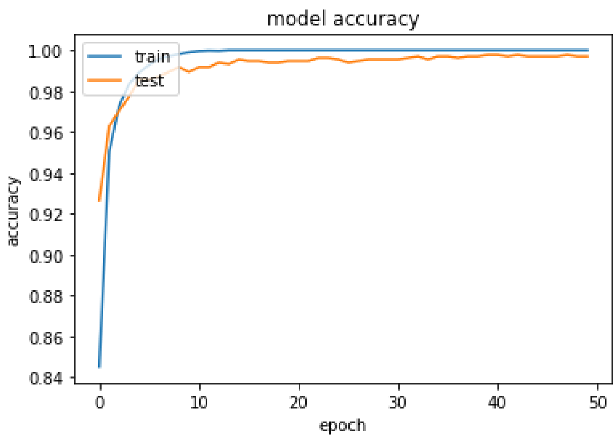
In [22]:

```
print(history.history.keys())
```

```
dict_keys(['val_loss', 'val_acc', 'loss', 'acc'])
```

In [23]:

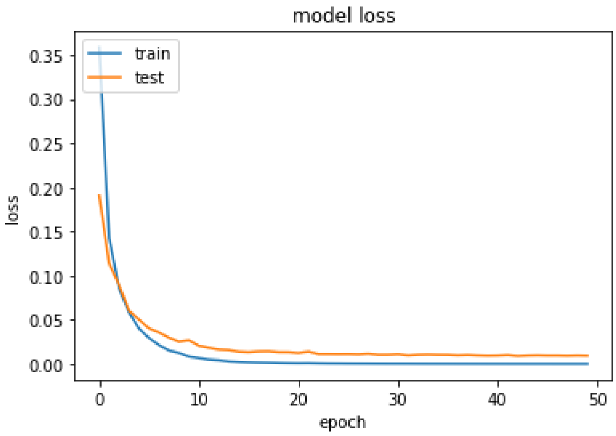
```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [24]:

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [25]:

```
#Reference is taken from here : https://machinelearningmastery.com/how-to-calculate-precision-reca
ll-f1-and-more-for-deep-learning-models/
# predict probabilities for test set
yhat_probs = model.predict(X_test, verbose=0)
# predict crisp classes for test set
yhat_classes = model.predict_classes(X_test, verbose=0)
```

In [26]:

```
# reduce to 1d array
yhat_probs = yhat_probs[:, 0]
yhat_classes = yhat_classes[:, 0]
```

In [27]:

```
accuracy = accuracy_score(y_test, yhat_classes)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_test, yhat_classes)
print('Precision: %f' % precision)

recall = recall_score(y_test, yhat_classes)
print('Recall: %f' % recall)

f1 = f1_score(y_test, yhat_classes)
print('F1 score: %f' % f1)
```

```
Accuracy: 0.996970
Precision: 0.995169
Recall: 0.985646
F1 score: 0.990385
```

Model 2

In [28]:

```
model_2 = Sequential()
model_2.add(Dense(128,activation='relu', input_shape=(166,)))
model_2.add(Dense(64,activation='relu'))
model_2.add(Dense(64,activation='relu'))
model_2.add(Dense(1,activation='sigmoid'))
```

In [29]:

```
model_2.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 128)               21376
_________________________________________________________________
dense_5 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_6 (Dense)              (None, 64)                4160
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 65
=================================================================
Total params: 33,857
Trainable params: 33,857
Non-trainable params: 0
_________________________________________________________________
```

In [30]:

```
model_2.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [31]:

```
history = model_2.fit(X_train_res,y_train_res,epochs=50,batch_size=128,validation_data=(X_test,y_te
st))
score1 = model_2.evaluate(X_test,y_test,verbose=0)
```

```
Train on 8940 samples, validate on 1320 samples
Epoch 1/50
8940/8940 [==============================] - 1s 104us/step - loss: 0.2601 - acc: 0.8909 -
val_loss: 0.1067 - val_acc: 0.9561
Epoch 2/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0717 - acc: 0.9762 - val_loss:
0.0725 - val_acc: 0.9765
Epoch 3/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0312 - acc: 0.9930 - val_loss:
0.0388 - val_acc: 0.9871
Epoch 4/50
8940/8940 [==============================] - 0s 26us/step - loss: 0.0155 - acc: 0.9964 - val_loss:
0.0319 - val_acc: 0.9924
Epoch 5/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0087 - acc: 0.9980 - val_loss:
0.0281 - val_acc: 0.9864
Epoch 6/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0056 - acc: 0.9985 - val_loss:
0.0159 - val_acc: 0.9932
Epoch 7/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0030 - acc: 0.9993 - val_loss:
0.0101 - val_acc: 0.9962
Epoch 8/50
8940/8940 [==============================] - 0s 28us/step - loss: 0.0018 - acc: 0.9996 - val_loss:
0.0143 - val_acc: 0.9955
Epoch 9/50
8940/8940 [==============================] - 0s 26us/step - loss: 8.5646e-04 - acc: 0.9999 - val_l
oss: 0.0142 - val_acc: 0.9955
Epoch 10/50
8940/8940 [==============================] - 0s 28us/step - loss: 4.0184e-04 - acc: 1.0000 - val_l
oss: 0.0112 - val_acc: 0.9955
Epoch 11/50
8940/8940 [==============================] - 0s 31us/step - loss: 3.0641e-04 - acc: 1.0000 - val_l
oss: 0.0107 - val_acc: 0.9955
Epoch 12/50
8940/8940 [==============================] - 0s 28us/step - loss: 2.3993e-04 - acc: 1.0000 - val_l
oss: 0.0110 - val_acc: 0.9955
Epoch 13/50
8940/8940 [==============================] - 0s 26us/step - loss: 1.9769e-04 - acc: 1.0000 - val_l
oss: 0.0099 - val_acc: 0.9955
Epoch 14/50
8940/8940 [==============================] - 0s 28us/step - loss: 1.6999e-04 - acc: 1.0000 - val_l
oss: 0.0090 - val_acc: 0.9947
Epoch 15/50
```

```
8940/8940 [==============================] - 0s 31us/step - loss: 1.4570e-04 - acc: 1.0000 - val_l
oss: 0.0094 - val_acc: 0.9947
Epoch 16/50
8940/8940 [==============================] - 0s 31us/step - loss: 1.2591e-04 - acc: 1.0000 - val_l
oss: 0.0099 - val_acc: 0.9970
Epoch 17/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.0882e-04 - acc: 1.0000 - val_l
oss: 0.0093 - val_acc: 0.9947
Epoch 18/50
8940/8940 [==============================] - 0s 28us/step - loss: 9.5328e-05 - acc: 1.0000 - val_l
oss: 0.0089 - val_acc: 0.9955
Epoch 19/50
8940/8940 [==============================] - 0s 30us/step - loss: 8.5975e-05 - acc: 1.0000 - val_l
oss: 0.0089 - val_acc: 0.9947
Epoch 20/50
8940/8940 [==============================] - 0s 30us/step - loss: 7.6997e-05 - acc: 1.0000 - val_l
oss: 0.0089 - val_acc: 0.9962
Epoch 21/50
8940/8940 [==============================] - 0s 26us/step - loss: 6.9161e-05 - acc: 1.0000 - val_l
oss: 0.0089 - val_acc: 0.9955
Epoch 22/50
8940/8940 [==============================] - 0s 28us/step - loss: 6.1239e-05 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9955
Epoch 23/50
8940/8940 [==============================] - 0s 30us/step - loss: 5.5789e-05 - acc: 1.0000 - val_l
oss: 0.0080 - val_acc: 0.9955
Epoch 24/50
8940/8940 [==============================] - 0s 28us/step - loss: 5.0465e-05 - acc: 1.0000 - val_l
oss: 0.0087 - val_acc: 0.9962
Epoch 25/50
8940/8940 [==============================] - 0s 28us/step - loss: 4.5418e-05 - acc: 1.0000 - val_l
oss: 0.0078 - val_acc: 0.9955
Epoch 26/50
8940/8940 [==============================] - 0s 30us/step - loss: 4.1619e-05 - acc: 1.0000 - val_l
oss: 0.0085 - val_acc: 0.9962
Epoch 27/50
8940/8940 [==============================] - 0s 33us/step - loss: 3.7573e-05 - acc: 1.0000 - val_l
oss: 0.0080 - val_acc: 0.9962
Epoch 28/50
8940/8940 [==============================] - 0s 28us/step - loss: 3.4856e-05 - acc: 1.0000 - val_l
oss: 0.0079 - val_acc: 0.9962
Epoch 29/50
8940/8940 [==============================] - 0s 28us/step - loss: 3.2247e-05 - acc: 1.0000 - val_l
oss: 0.0077 - val_acc: 0.9955
Epoch 30/50
8940/8940 [==============================] - 0s 30us/step - loss: 2.9669e-05 - acc: 1.0000 - val_l
oss: 0.0078 - val_acc: 0.9962
Epoch 31/50
8940/8940 [==============================] - 0s 31us/step - loss: 2.7471e-05 - acc: 1.0000 - val_l
oss: 0.0076 - val_acc: 0.9955
Epoch 32/50
8940/8940 [==============================] - 0s 31us/step - loss: 2.4946e-05 - acc: 1.0000 - val_l
oss: 0.0082 - val_acc: 0.9962
Epoch 33/50
8940/8940 [==============================] - 0s 31us/step - loss: 2.3264e-05 - acc: 1.0000 - val_l
oss: 0.0074 - val_acc: 0.9962
Epoch 34/50
8940/8940 [==============================] - 0s 28us/step - loss: 2.1972e-05 - acc: 1.0000 - val_l
oss: 0.0081 - val_acc: 0.9962
Epoch 35/50
8940/8940 [==============================] - 0s 30us/step - loss: 2.0005e-05 - acc: 1.0000 - val_l
oss: 0.0075 - val_acc: 0.9962
Epoch 36/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.8838e-05 - acc: 1.0000 - val_l
oss: 0.0074 - val_acc: 0.9962
Epoch 37/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.7737e-05 - acc: 1.0000 - val_l
oss: 0.0076 - val_acc: 0.9970
Epoch 38/50
8940/8940 [==============================] - 0s 31us/step - loss: 1.6489e-05 - acc: 1.0000 - val_l
oss: 0.0077 - val_acc: 0.9962
Epoch 39/50
8940/8940 [==============================] - 0s 31us/step - loss: 1.5102e-05 - acc: 1.0000 - val_l
oss: 0.0077 - val_acc: 0.9962
Epoch 40/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.4171e-05 - acc: 1.0000 - val_l
oss: 0.0078 - val_acc: 0.9962
```

```
Epoch 41/50
8940/8940 [==============================] - 0s 33us/step - loss: 1.3309e-05 - acc: 1.0000 - val_l
oss: 0.0077 - val_acc: 0.9970
Epoch 42/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.2470e-05 - acc: 1.0000 - val_l
oss: 0.0076 - val_acc: 0.9970
Epoch 43/50
8940/8940 [==============================] - 0s 30us/step - loss: 1.1589e-05 - acc: 1.0000 - val_l
oss: 0.0075 - val_acc: 0.9970
Epoch 44/50
8940/8940 [==============================] - 0s 31us/step - loss: 1.1036e-05 - acc: 1.0000 - val_l
oss: 0.0075 - val_acc: 0.9970
Epoch 45/50
8940/8940 [==============================] - 0s 33us/step - loss: 1.0188e-05 - acc: 1.0000 - val_l
oss: 0.0076 - val_acc: 0.9970
Epoch 46/50
8940/8940 [==============================] - 0s 30us/step - loss: 9.6192e-06 - acc: 1.0000 - val_l
oss: 0.0078 - val_acc: 0.9962
Epoch 47/50
8940/8940 [==============================] - 0s 28us/step - loss: 8.9732e-06 - acc: 1.0000 - val_l
oss: 0.0078 - val_acc: 0.9970
Epoch 48/50
8940/8940 [==============================] - 0s 31us/step - loss: 8.4417e-06 - acc: 1.0000 - val_l
oss: 0.0074 - val_acc: 0.9970
Epoch 49/50
8940/8940 [==============================] - 0s 30us/step - loss: 7.8780e-06 - acc: 1.0000 - val_l
oss: 0.0079 - val_acc: 0.9970
Epoch 50/50
8940/8940 [==============================] - 0s 28us/step - loss: 7.5853e-06 - acc: 1.0000 - val_l
oss: 0.0077 - val_acc: 0.9970
```

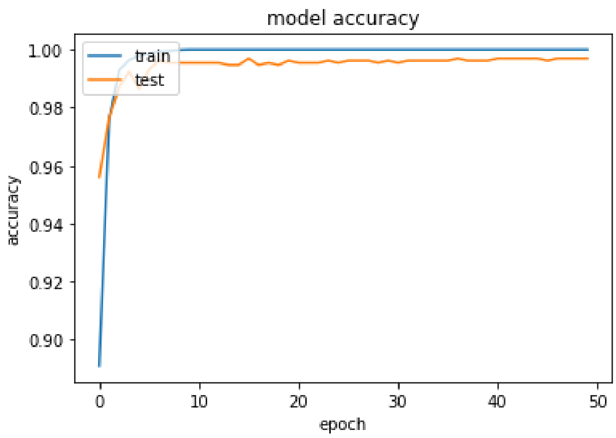
In [32]:

```
print("Accuracy : %.2f%%" % (score1[1]*100))
```

```
Accuracy : 99.70%
```

In [34]:

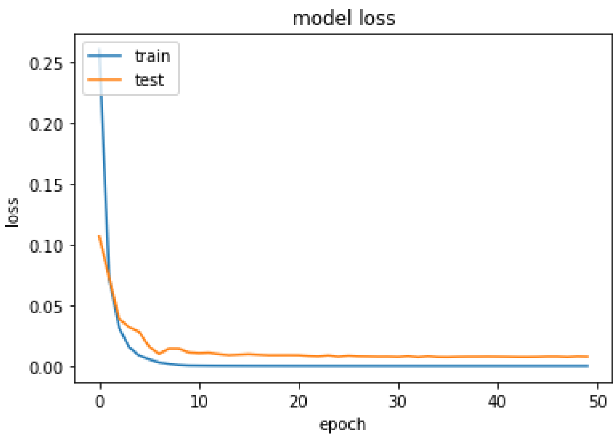
```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [35]:

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [36]:

```
# predict probabilities for test set
yhat_probs = model_2.predict(X_test, verbose=0)
# predict crisp classes for test set
yhat_classes = model_2.predict_classes(X_test, verbose=0)
```

In [37]:

```
# reduce to 1d array
yhat_probs = yhat_probs[:, 0]
yhat_classes = yhat_classes[:, 0]
```

In [38]:

```
accuracy = accuracy_score(y_test, yhat_classes)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_test, yhat_classes)
print('Precision: %f' % precision)

recall = recall_score(y_test, yhat_classes)
print('Recall: %f' % recall)

f1 = f1_score(y_test, yhat_classes)
print('F1 score: %f' % f1)
```

```
Accuracy: 0.996970
Precision: 0.995169
Recall: 0.985646
F1 score: 0.990385
```

Model 3

In [39]:

```
model_3 = Sequential()
model_3.add(Dense(264,activation='relu', input_shape=(166,)))
model_3.add(Dense(128,activation='relu'))
model_3.add(Dense(128,activation='relu'))
model_3.add(Dense(64,activation='relu'))
model_3.add(Dense(1,activation='sigmoid'))
```

In [40]:

```
model_3.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_8 (Dense)              (None, 264)               44088     
_________________________________________________________________
dense_9 (Dense)              (None, 128)               33920     
_________________________________________________________________
dense_10 (Dense)             (None, 128)               16512     
_________________________________________________________________
dense_11 (Dense)             (None, 64)                8256      
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 65        
=================================================================
Total params: 102,841
Trainable params: 102,841
Non-trainable params: 0
_________________________________________________________________
```

In [41]:

```
model_3.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

In [42]:

```
history = model_3.fit(X_train_res,y_train_res,epochs=50,batch_size=128,validation_data=(X_test,y_te
st))
score3 = model_3.evaluate(X_test,y_test,verbose=0)
```

```
Train on 8940 samples, validate on 1320 samples
Epoch 1/50
8940/8940 [==============================] - 1s 132us/step - loss: 0.2194 - acc: 0.9030 -
val_loss: 0.0942 - val_acc: 0.9591
Epoch 2/50
8940/8940 [==============================] - 0s 38us/step - loss: 0.0544 - acc: 0.9776 - val_loss:
0.0514 - val_acc: 0.9795
Epoch 3/50
8940/8940 [==============================] - 0s 40us/step - loss: 0.0289 - acc: 0.9896 - val_loss:
0.0354 - val_acc: 0.9833
Epoch 4/50
8940/8940 [==============================] - 0s 40us/step - loss: 0.0099 - acc: 0.9969 - val_loss:
0.0192 - val_acc: 0.9947
Epoch 5/50
8940/8940 [==============================] - 0s 40us/step - loss: 0.0056 - acc: 0.9984 - val_loss:
0.0183 - val_acc: 0.9924
Epoch 6/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0054 - acc: 0.9983 - val_loss:
0.0110 - val_acc: 0.9947
Epoch 7/50
8940/8940 [==============================] - 0s 38us/step - loss: 0.0024 - acc: 0.9992 - val_loss:
0.0248 - val_acc: 0.9939
Epoch 8/50
8940/8940 [==============================] - 0s 40us/step - loss: 0.0196 - acc: 0.9932 - val_loss:
0.0153 - val_acc: 0.9924
Epoch 9/50
8940/8940 [==============================] - 0s 40us/step - loss: 0.0024 - acc: 0.9992 - val_loss:
0.0103 - val_acc: 0.9962
Epoch 10/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0019 - acc: 0.9993 - val_loss:
0.0113 - val_acc: 0.9977
Epoch 11/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0014 - acc: 0.9996 - val_loss:
0.0085 - val_acc: 0.9962
Epoch 12/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0044 - acc: 0.9984 - val_loss:
0.0124 - val_acc: 0.9962
Epoch 13/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0076 - acc: 0.9985 - val_loss:
0.9087 - val_acc: 0.9000
Epoch 14/50
8940/8940 [==============================] - 0s 44us/step - loss: 0.0688 - acc: 0.9885 - val_loss:
0.0149 - val_acc: 0.9939
Epoch 15/50
8940/8940 [==============================] - 0s 42us/step - loss: 0.0094 - acc: 0.9969 - val_loss:
```

```
0.0176 - val_acc: 0.9932
Epoch 16/50
8940/8940 [==============================] - 0s 44us/step - loss: 0.0022 - acc: 0.9996 - val_loss:
0.0073 - val_acc: 0.9970
Epoch 17/50
8940/8940 [==============================] - 0s 44us/step - loss: 4.6419e-04 - acc: 1.0000 - val_l
oss: 0.0068 - val_acc: 0.9962
Epoch 18/50
8940/8940 [==============================] - 0s 45us/step - loss: 8.2755e-05 - acc: 1.0000 - val_l
oss: 0.0076 - val_acc: 0.9970
Epoch 19/50
8940/8940 [==============================] - 0s 40us/step - loss: 4.5203e-05 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9970
Epoch 20/50
8940/8940 [==============================] - 0s 42us/step - loss: 2.8175e-05 - acc: 1.0000 - val_l
oss: 0.0082 - val_acc: 0.9962
Epoch 21/50
8940/8940 [==============================] - 0s 47us/step - loss: 2.0245e-05 - acc: 1.0000 - val_l
oss: 0.0079 - val_acc: 0.9962
Epoch 22/50
8940/8940 [==============================] - 0s 44us/step - loss: 1.5590e-05 - acc: 1.0000 - val_l
oss: 0.0082 - val_acc: 0.9962
Epoch 23/50
8940/8940 [==============================] - 0s 42us/step - loss: 1.2363e-05 - acc: 1.0000 - val_l
oss: 0.0082 - val_acc: 0.9962
Epoch 24/50
8940/8940 [==============================] - 0s 45us/step - loss: 1.0076e-05 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9962
Epoch 25/50
8940/8940 [==============================] - 0s 47us/step - loss: 8.4602e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 26/50
8940/8940 [==============================] - 0s 42us/step - loss: 7.1334e-06 - acc: 1.0000 - val_l
oss: 0.0085 - val_acc: 0.9962
Epoch 27/50
8940/8940 [==============================] - 0s 47us/step - loss: 6.1366e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 28/50
8940/8940 [==============================] - 0s 44us/step - loss: 5.3270e-06 - acc: 1.0000 - val_l
oss: 0.0085 - val_acc: 0.9962
Epoch 29/50
8940/8940 [==============================] - 0s 45us/step - loss: 4.6611e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 30/50
8940/8940 [==============================] - 0s 45us/step - loss: 4.1148e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 31/50
8940/8940 [==============================] - 0s 44us/step - loss: 3.6581e-06 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9962
Epoch 32/50
8940/8940 [==============================] - 0s 45us/step - loss: 3.2791e-06 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9962
Epoch 33/50
8940/8940 [==============================] - 0s 44us/step - loss: 2.9431e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 34/50
8940/8940 [==============================] - 0s 42us/step - loss: 2.6647e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 35/50
8940/8940 [==============================] - 0s 47us/step - loss: 2.4210e-06 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9962
Epoch 36/50
8940/8940 [==============================] - 0s 44us/step - loss: 2.2049e-06 - acc: 1.0000 - val_l
oss: 0.0083 - val_acc: 0.9962
Epoch 37/50
8940/8940 [==============================] - 0s 51us/step - loss: 2.0265e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 38/50
8940/8940 [==============================] - 0s 42us/step - loss: 1.8555e-06 - acc: 1.0000 - val_l
oss: 0.0085 - val_acc: 0.9962
Epoch 39/50
8940/8940 [==============================] - 0s 45us/step - loss: 1.7182e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 40/50
8940/8940 [==============================] - 0s 49us/step - loss: 1.5795e-06 - acc: 1.0000 - val_l
oss: 0.0084 - val_acc: 0.9962
Epoch 41/50
```

```
8940/8940 [==============================] - 0s 40us/step - loss: 1.4694e-06 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 42/50
8940/8940 [==============================] - 0s 44us/step - loss: 1.3660e-06 - acc: 1.0000 - val_l
oss: 0.0085 - val_acc: 0.9962
Epoch 43/50
8940/8940 [==============================] - 0s 44us/step - loss: 1.2678e-06 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 44/50
8940/8940 [==============================] - 0s 44us/step - loss: 1.1855e-06 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 45/50
8940/8940 [==============================] - 0s 47us/step - loss: 1.1080e-06 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 46/50
8940/8940 [==============================] - 0s 44us/step - loss: 1.0376e-06 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 47/50
8940/8940 [==============================] - 0s 45us/step - loss: 9.7341e-07 - acc: 1.0000 - val_l
oss: 0.0086 - val_acc: 0.9962
Epoch 48/50
8940/8940 [==============================] - 0s 42us/step - loss: 9.1459e-07 - acc: 1.0000 - val_l
oss: 0.0087 - val_acc: 0.9962
Epoch 49/50
8940/8940 [==============================] - 0s 44us/step - loss: 8.6291e-07 - acc: 1.0000 - val_l
oss: 0.0088 - val_acc: 0.9962
Epoch 50/50
8940/8940 [==============================] - 0s 44us/step - loss: 8.1299e-07 - acc: 1.0000 - val_l
oss: 0.0087 - val_acc: 0.9962
```

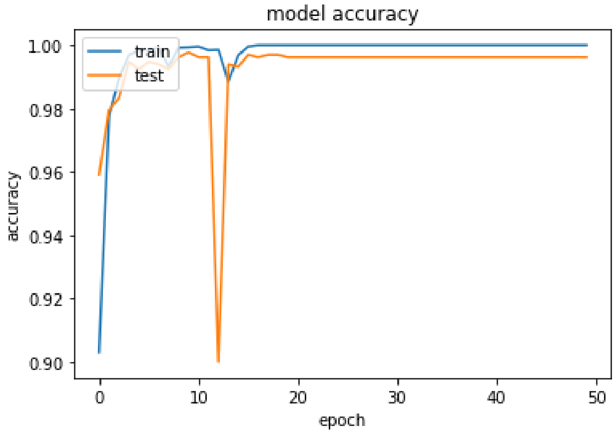
In [43]:

```
print("Accuracy : %.2f%%" % (score3[1]*100))
```

```
Accuracy : 99.62%
```

In [44]:

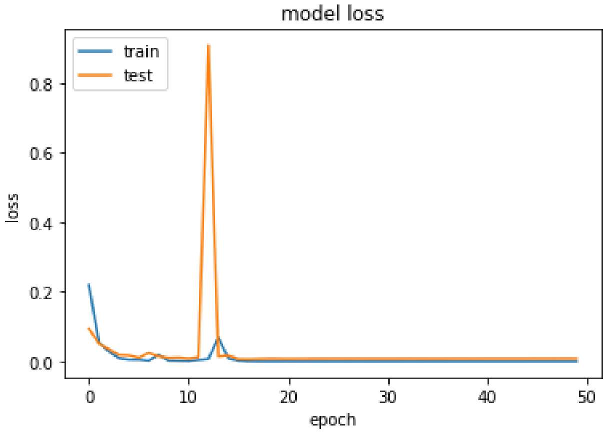
```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [45]:

```
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
```

```
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



In [46]:

```
# predict probabilities for test set
yhat_probs = model_3.predict(X_test, verbose=0)
# predict crisp classes for test set
yhat_classes = model_3.predict_classes(X_test, verbose=0)
```

In [47]:

```
# reduce to 1d array
yhat_probs = yhat_probs[:, 0]
yhat_classes = yhat_classes[:, 0]
```

In [48]:

```
accuracy = accuracy_score(y_test, yhat_classes)
print('Accuracy: %f' % accuracy)

precision = precision_score(y_test, yhat_classes)
print('Precision: %f' % precision)

recall = recall_score(y_test, yhat_classes)
print('Recall: %f' % recall)

f1 = f1_score(y_test, yhat_classes)
print('F1 score: %f' % f1)
```

```
Accuracy: 0.996212
Precision: 0.990385
Recall: 0.985646
F1 score: 0.988010
```

Prettytable

In [49]:

```
from prettytable import PrettyTable
x=PrettyTable()
print("Machine Learning Models")
x.field_names=['Model','Accuracy','Precision','Recall','F1-Score']
x.add_row(['Model 1',0.996970,0.995169,0.985646,0.990385])
x.add_row(['Model 2',0.996970,0.995169,0.985646,0.990385])
x.add_row(['Model 3',0.996212,0.990385,0.985646,0.988010])
print(x)
```

```
Machine Learning Models
+---------+----------+-----------+----------+----------+
|  Model  | Accuracy | Precision |  Recall  | F1-Score |
+---------+----------+-----------+----------+----------+
| Model 1 | 0.99697  |  0.995169 | 0.985646 | 0.990385 |
```

```
| Model 1 | 0.99697  |  0.995169 | 0.985646 | 0.990385 |
| Model 2 | 0.99697  |  0.995169 | 0.985646 | 0.990385 |
| Model 3 | 0.996212 |  0.990385 | 0.985646 | 0.98801  |
+---------+----------+-----------+----------+----------+
```

Conclusions

1. This problem is a binary class classification problem and we have to predict class for organic chemical compounds.
2. Total features in the dataset are 170 which includes target feature class.
3. molecule_name, conformation_name and ID are not important features for prediction so we drop them.
4. We have build three models. Each model is a Multi Layered Perceptron model. MLPs are suitable for classification prediction problems where inputs are assigned a class or label
5. Each model is sequential model which is a linear stack of Layers.
6. We have plotted Accuracy and Loss graphs for each model.
7. Also calculated Acccuracy, Precision, Recall and F1-score values for each model.
8. Class feature of Dataset was imbalanced which effects the model. We used SMOTE technique to balance the dataset so that model does not do predictions on the basis of majority class.
9. For preprocessing we did Data Standardization to scale the data.
10. There were no null values in the data.
11. Also size of the dataset is small. That's why accuracy of model is almost 100%. We should have more data for model to perform better.
12. Model 1 and 2 have less layers than Model 3 and recall value is same for all models.
13. I have used Model 1 for predictions.

In [ ]:

```

```