In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import accuracy_score
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report,confusion_matrix
```

In [2]:
```
# Import data set csv
#data_set = pd.read_csv('filtered_data_set.csv')
data_set = pd.read_csv('expanded_filtered_data_set.csv')
```

In [3]:
```
# Set X and y columns
X = data_set[['valence','acousticness','danceability','duration_ms','energy','exp
              'instrumentalness','key','liveness','loudness','mode','speechiness',
y = data_set['popularity'].values
```

In [4]:
```
# Create the X training and testing set, and Y training and testing set where 70%
# are for the training set and the rest to the testing set.
x_testing_set, x_training_set, y_testing_set, y_training_set = train_test_split()
                                                                              s
```

In [5]:
```
# Create model we still construct sequentially
model = Sequential()

# Add dense (every input connected to all units in hidden layer)
# Activation - sigmoid maps between 0 and 1. relu maps to 0 or 1
model.add(Dense(15, input_dim=13, activation='relu'))
model.add(Dense(18, activation='relu'))
model.add(Dense(13, activation='relu'))
model.add(Dense(10, activation='relu'))

# Output layer
model.add(Dense(1, activation='sigmoid'))
```

```
WARNING:tensorflow:From D:\Anaconda\lib\site-packages\tensorflow\python\ops\ini
t_ops.py:1251: calling VarianceScaling.__init__ (from tensorflow.python.ops.ini
t_ops) with dtype is deprecated and will be removed in a future version.
Instructions for updating:
Call initializer instance with the dtype argument instead of passing it to the
constructor
```

In [6]:
```
# Compile the model.
# Optimizer - Adam is an efficient optimize to apply gradient descent to the mode
# Metrics - want the accuracy on how the model predicts
model.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])
```

```
WARNING:tensorflow:From D:\Anaconda\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 [7]:
```
model.fit(x_training_set,y_training_set,epochs=100, batch_size=64)
```

```
94188/94188 [==============================] - 1s 12us/sample - loss: 0.4283
- acc: 0.8076
Epoch 96/100
94188/94188 [==============================] - 1s 11us/sample - loss: 0.4283
- acc: 0.8078
Epoch 97/100
94188/94188 [==============================] - 1s 11us/sample - loss: 0.4282
- acc: 0.8075
Epoch 98/100
94188/94188 [==============================] - 1s 11us/sample - loss: 0.4278
- acc: 0.8079
Epoch 99/100
94188/94188 [==============================] - 1s 12us/sample - loss: 0.4285
- acc: 0.8074
Epoch 100/100
94188/94188 [==============================] - 1s 12us/sample - loss: 0.4278
- acc: 0.8071
```

Out[7]:
```
<tensorflow.python.keras.callbacks.History at 0x2c5469f1dc8>
```

In [8]:
```
# Get the predicted values with the testing set
test_predictions = model.predict(x_testing_set)
```

In [10]:
```
# Resize to series
#test_predictions = pd.Series(test_predictions.reshape(2961,))
test_predictions = pd.Series(test_predictions.reshape(40365,))
```

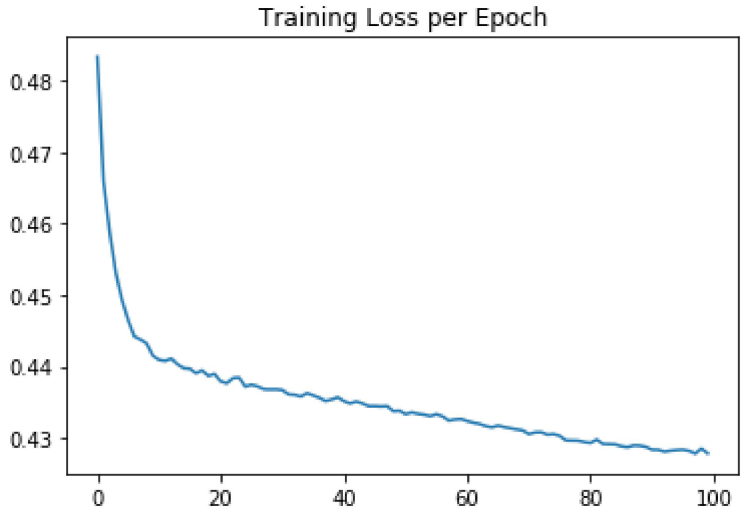
In [11]:
```
training_score = model.evaluate(x_training_set,y_training_set)
test_score = model.evaluate(x_testing_set,y_testing_set)
print(training_score)
print(test_score)
```

```
94188/94188 [==============================] - 1s 10us/sample - loss: 0.4248 -
acc: 0.8100
40365/40365 [==============================] - 0s 11us/sample - loss: 0.4323 -
acc: 0.8041
[0.4248153982315316, 0.8099758]
[0.43227773612499176, 0.8040629]
```

In [12]:
```
# Find predict y values with the x testing set and find accuracy
ynew = model.predict_classes(x_testing_set)
correct=0
for i in range(0,len(ynew)):
    if(ynew[i]==y_testing_set[i]):
        correct = correct + 1
print("Accuracy=", correct/len(test_predictions))
```

```
Accuracy= 0.8040629258020562
```

In [13]:
```
loss = model.history.history['loss']
sns.lineplot(x=range(len(loss)),y=loss)
plt.title("Training Loss per Epoch");
```



In [14]:
```
print(classification_report(y_testing_set, ynew))
```

```
              precision    recall  f1-score   support

         0.0       0.82      0.94      0.88     30869
         1.0       0.66      0.35      0.46      9496

    accuracy                           0.80     40365
   macro avg       0.74      0.65      0.67     40365
weighted avg       0.79      0.80      0.78     40365
```

In [ ]:
```

```