Greg Walsh

DSSA-5104-091 - DEEP LEARNING

Spring 2020

Voting records - party prediction

Neural Network for binary classification

In [1]:
```
# Import necessary libraries
from keras.models import Sequential
from keras.layers import Dense
from keras import optimizers
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt

# Load in Data
dataset = pd.read_csv("votingrecords.csv", header = None)

# set random seed for reproducibility
np.random.seed(7)
```

```
Using TensorFlow backend.
```

In [2]:
```
# Lets look at the data
dataset.head()
```

Out[2]:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	democrat	n	y	y	n	y	y	n	n	n	n	n	n	y	y	y	y
1	republican	n	y	n	y	y	y	n	n	n	n	n	y	y	y	n	y
2	democrat	y	y	y	n	n	n	y	y	y	n	y	n	n	n	y	y
3	democrat	y	y	y	n	n	n	y	y	y	n	n	n	n	n	y	y
4	democrat	y	n	y	n	n	n	y	y	y	y	n	n	n	n	y	y

In [4]:
```
# Our data is y's and n's and a NN needs 1's and 0's so were going to us
e LabelEncoder()
data = dataset.apply(LabelEncoder().fit_transform)

# Convert dataframe into a numpy array
df = data.values

print(df)

# Shows # of observations and variables
print(df.shape)
```

```
[[0 0 1 ... 1 1 1]
 [1 0 1 ... 1 0 1]
 [0 1 1 ... 0 1 1]
 ...
 [1 0 0 ... 1 0 1]
 [1 0 0 ... 1 0 1]
 [0 0 0 ... 0 0 1]]
(232, 17)
```

In [6]:
```
# This is a much better format. Now we can split into our X and Y Variab
les
# X is our input
# Y is our output
X = df[:,1:17]
Y = df[:,0]
```

In [7]:
```
# We want to look into how the NN can be changed so we are going to cap
 our epochs at 50 for all tests
# Testing with anything higher than 150 give 100% accuracy or very close
so we need a very low epoch
```

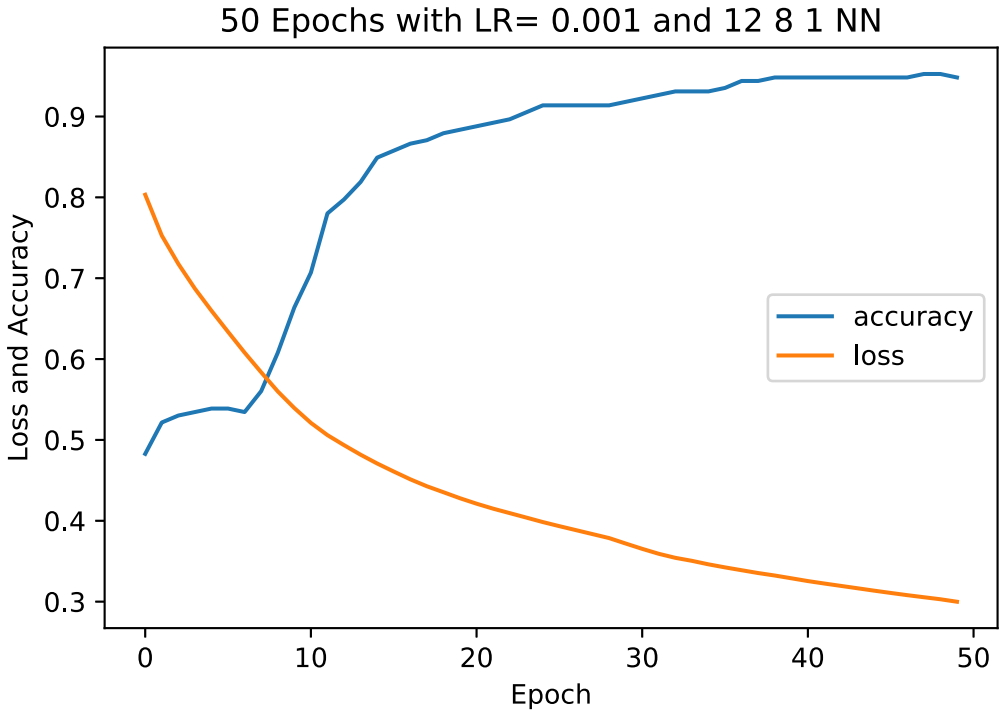
In [8]:
```
# 3 Layers Relu Relu Sigmoid 12 8 1 NN
# Reset Variables
model = None
history = None
Y_predict = None
adam = None

# Set Epochs
epochs = 50
# Create NN Model
model = Sequential()
model.add(Dense(12, input_dim = 16, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
# Compile the NN
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac
curacy'])

# Fit the NN 50 with epochs
history = model.fit(X,Y,epochs=epochs, verbose=0)

# Evaluate the NN
scores = model.evaluate(X, Y)

# Predict the NN
Y_predict = model.predict(X)

# Print our Accuracy and Loss
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
```

```
232/232 [==============================] - 0s 1ms/step

accuracy: 95.26%

loss: 29.80%
```

In [9]:
```
# I have included this code for you which will
# create confusion matrix details
rounded = [round(i[0]) for i in Y_predict]
y_pred = np.array(rounded,dtype='int64')
print('================')
print('Confusion Matrix')
print('================')
CM = confusion_matrix(Y, y_pred)
print('True negatives: ',  CM[0,0])
print('False negatives: ', CM[1,0])
print('False positives: ', CM[0,1])
print('True positives: ',  CM[1,1])
```

```
================
Confusion Matrix
================
True negatives:  121
False negatives:  8
False positives:  3
True positives:  100
```

In [10]:
```
# Plot Loss and Accuracy by Epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
plt.title('50 Epochs with LR= 0.001 and 12 8 1 NN')
plt.ylabel('Loss and Accuracy')
plt.xlabel('Epoch')
plt.legend(['accuracy', 'loss'], loc='center right')
plt.text(15,-0.2, 'Results in 97.84% Accuracy and 6.56% Loss', ha='cente
r', va='top')
plt.show()
```



Results in 97.84% Accuracy and 6.56% Loss

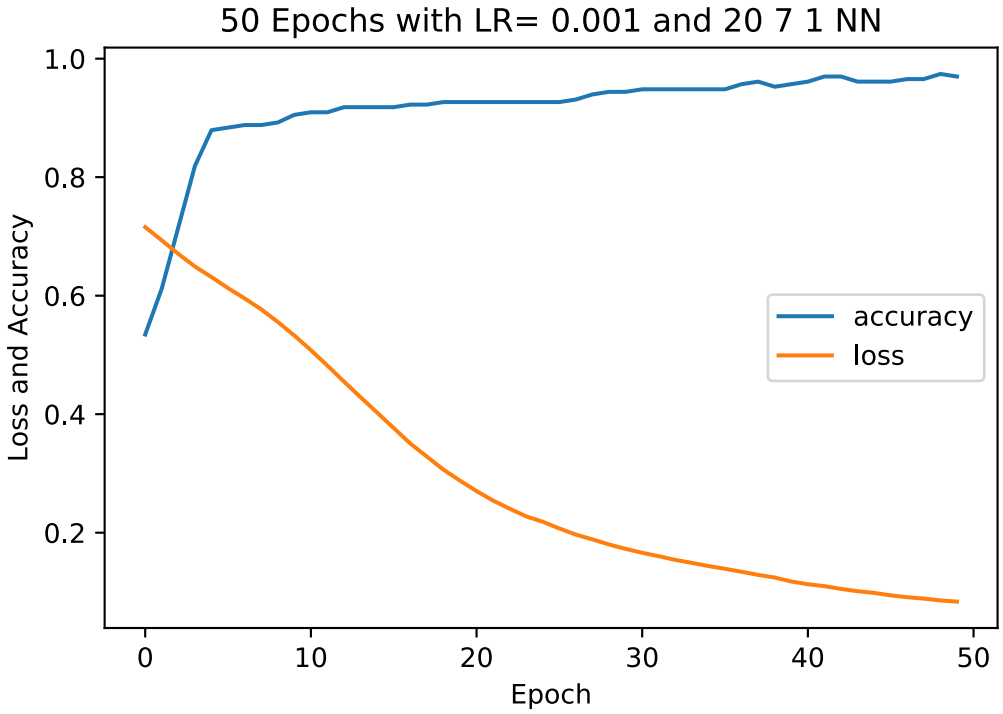
In [11]:
```
# 3 Layers Relu Relu Sigmoid 20 7 1 NN
# Reset Variables
model = None
history = None
Y_predict = None
adam = None
# Create NN Model
model = Sequential()
model.add(Dense(20, input_dim = 16, activation='relu'))
model.add(Dense(7, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
# Compile the NN
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac
curacy'])

# Fit the NN 50 with epochs
history = model.fit(X,Y,epochs=epochs,verbose=0)

# Evaluate the NN
scores = model.evaluate(X, Y)

# Predict the NN
Y_predict = model.predict(X)

# Print our Accuracy and Loss
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
```

```
232/232 [==============================] - 0s 873us/step

accuracy: 96.98%

loss: 8.22%
```

In [50]:
```
# I have included this code for you which will
# create confusion matrix details
rounded = [round(i[0]) for i in Y_predict]
y_pred = np.array(rounded,dtype='int64')
print('================')
print('Confusion Matrix')
print('================')
CM = confusion_matrix(Y, y_pred)
print('True negatives: ',  CM[0,0])
print('False negatives: ', CM[1,0])
print('False positives: ', CM[0,1])
print('True positives: ',  CM[1,1])
```

```
================
Confusion Matrix
================
True negatives:  121
False negatives:  2
False positives:  3
True positives:  106
```

In [12]:
```
# Plot Loss and Accuracy by Epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
plt.title('50 Epochs with LR= 0.001 and 20 7 1 NN')
plt.ylabel('Loss and Accuracy')
plt.xlabel('Epoch')
plt.legend(['accuracy', 'loss'], loc='center right')
plt.text(15,-0.2, 'Results in 99.14% Accuracy and 3.08% Loss', ha='cente
r', va='top')
plt.show()
```



Results in 99.14% Accuracy and 3.08% Loss

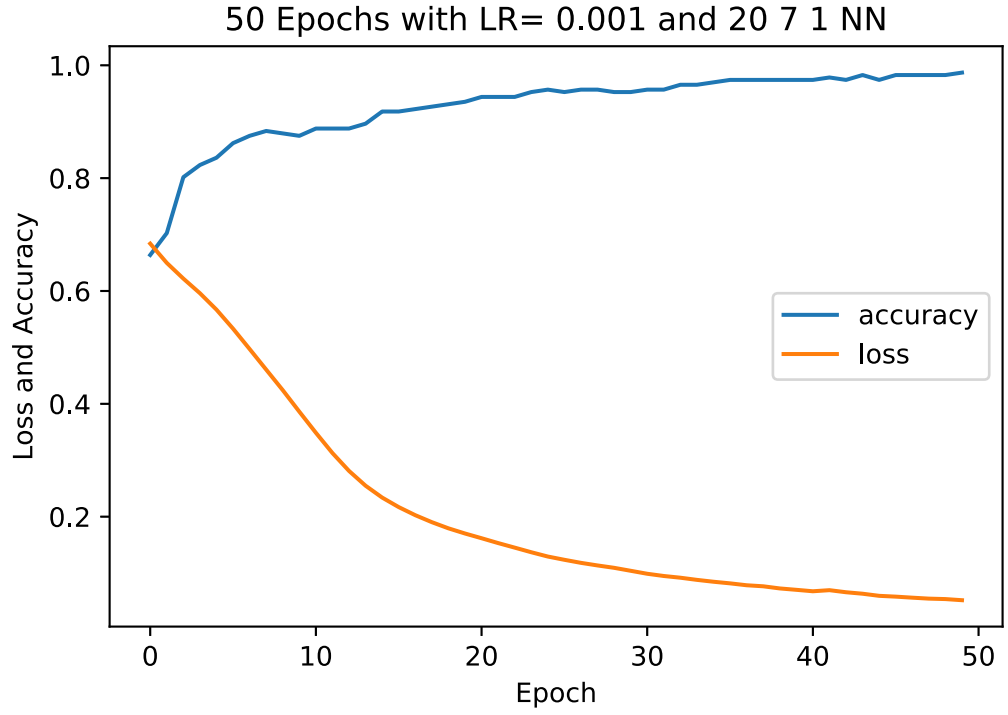
In [13]:
```
# 4 Layers Relu Relu Relu Sigmoid 5 20 10 1 NN
# Reset Variables
model = None
history = None
Y_predict = None
adam = None
# Create NN Model
model = Sequential()
model.add(Dense(5, input_dim = 16, activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
# Compile the NN
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac
curacy'])

# Fit the NN 50 with epochs
history = model.fit(X,Y,epochs=epochs, verbose=0)

# Evaluate the NN
scores = model.evaluate(X, Y)

# Predict the NN
Y_predict = model.predict(X)

# Print our Accuracy and Loss
print(epochs)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
```

```
232/232 [==============================] - 0s 789us/step
50

accuracy: 98.71%

loss: 5.17%
```

In [14]:
```
# I have included this code for you which will
# create confusion matrix details
rounded = [round(i[0]) for i in Y_predict]
y_pred = np.array(rounded,dtype='int64')
print('================')
print('Confusion Matrix')
print('================')
CM = confusion_matrix(Y, y_pred)
print('True negatives: ',  CM[0,0])
print('False negatives: ', CM[1,0])
print('False positives: ', CM[0,1])
print('True positives: ',  CM[1,1])
```

```
================
Confusion Matrix
================
True negatives:  122
False negatives:  1
False positives:  2
True positives:  107
```

In [15]:
```
# Plot Loss and Accuracy by Epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
plt.title('50 Epochs with LR= 0.001 and 5 20 10 1 NN')
plt.ylabel('Loss and Accuracy')
plt.xlabel('Epoch')
plt.legend(['accuracy', 'loss'], loc='center right')
plt.text(15,-0.2, 'Results in 98.71% Accuracy and 5.17% Loss', ha='cente
r', va='top')
splt.show()
```



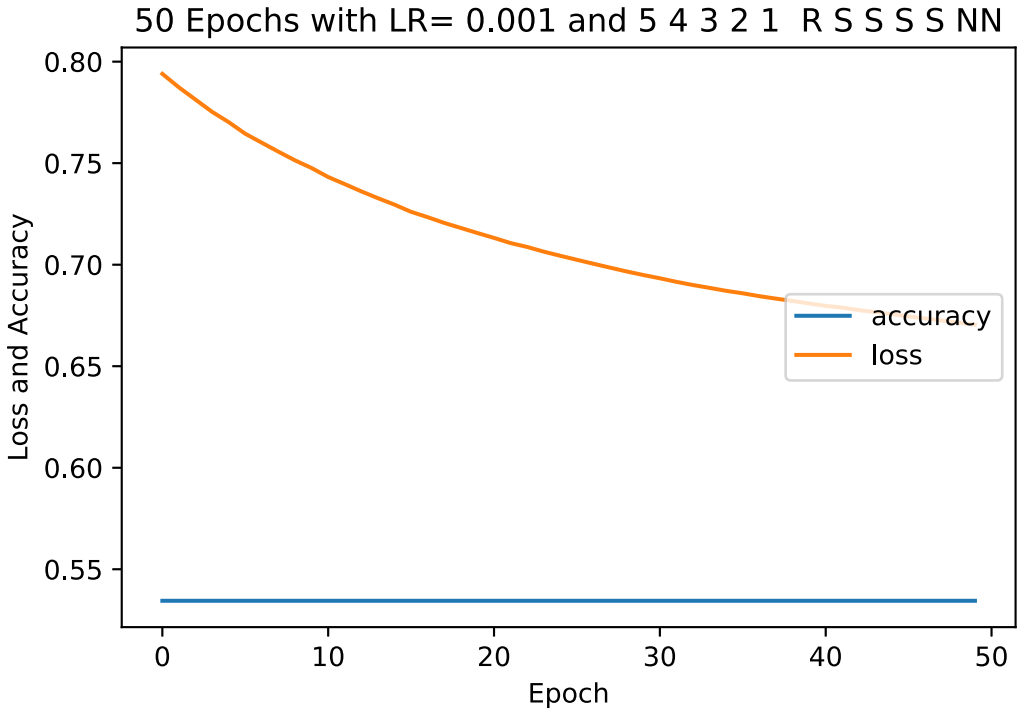
In [26]:
```
# 5 Layers Relu Sigmoid Sigmoid Sigmoid 5 4 3 2 1 NN
# Reset Variables
model = None
history = None
Y_predict = None
adam = None
# Create NN Model
model = Sequential()
model.add(Dense(5, input_dim = 16, activation='relu'))
model.add(Dense(4, activation='sigmoid'))
model.add(Dense(3, activation='sigmoid'))
model.add(Dense(2, activation='sigmoid'))
model.add(Dense(1, activation='sigmoid'))



adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
# Compile the NN
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac
curacy'])
# Fit the NN 50 with epochs
history = model.fit(X,Y,epochs=epochs, verbose=0)
# Evaluate the NN
scores = model.evaluate(X, Y)
# Predict the NN
Y_predict = model.predict(X)
# Print our Accuracy and Loss
print(epochs)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
```

```
232/232 [==============================] - 0s 864us/step
50

accuracy: 53.45%

loss: 66.99%
```

In [27]:
```
# I have included this code for you which will
# create confusion matrix details
rounded = [round(i[0]) for i in Y_predict]
y_pred = np.array(rounded,dtype='int64')
print('================')
print('Confusion Matrix')
print('================')
CM = confusion_matrix(Y, y_pred)
print('True negatives: ',  CM[0,0])
print('False negatives: ', CM[1,0])
print('False positives: ', CM[0,1])
print('True positives: ',  CM[1,1])
```

```
================
Confusion Matrix
================
True negatives:  124
False negatives:  108
False positives:  0
True positives:  0
```

In [42]:
```
# Plot Loss and Accuracy by Epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
plt.title('50 Epochs with LR= 0.001 and 5 4 3 2 1  R S S S S NN')
plt.ylabel('Loss and Accuracy')
plt.xlabel('Epoch')
plt.legend(['accuracy', 'loss'], loc='center right')
plt.text(15,.4, 'Results in 53.45% Accuracy and 66.99% Loss', ha='cente
r', va='top')
plt.show()
```



Results in 53.45% Accuracy and 66.99% Loss

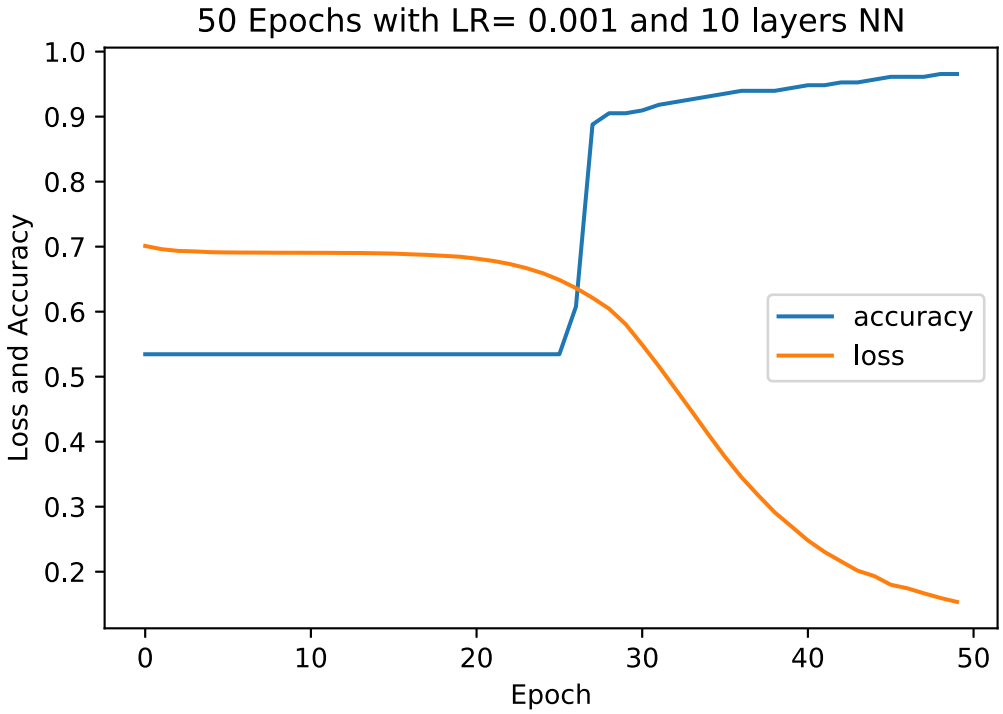
In [45]:
```
# 10 Layers
# Reset Variables
model = None
history = None
Y_predict = None
adam = None
# Create NN Model
model = Sequential()
model.add(Dense(8, input_dim = 16, activation='relu'))
model.add(Dense(7, activation='sigmoid'))
model.add(Dense(10, activation='relu'))
model.add(Dense(9, activation='sigmoid'))
model.add(Dense(4, activation='relu'))
model.add(Dense(8, activation='sigmoid'))
model.add(Dense(12, activation='relu'))
model.add(Dense(9, activation='sigmoid'))
model.add(Dense(10, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
adam = optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None,
decay=0.0, amsgrad=False)
# Compile the NN
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['ac
curacy'])

# Fit the NN 50 with epochs
history = model.fit(X,Y,epochs=epochs, verbose=0)

# Evaluate the NN
scores = model.evaluate(X, Y)

# Predict the NN
Y_predict = model.predict(X)

# Print our Accuracy and Loss
print(epochs)
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
print("\n%s: %.2f%%" % (model.metrics_names[0], scores[0]*100))
```

```
232/232 [==============================] - 0s 1ms/step
50

accuracy: 96.55%

loss: 14.86%
```

In [46]:
```
# I have included this code for you which will
# create confusion matrix details
rounded = [round(i[0]) for i in Y_predict]
y_pred = np.array(rounded,dtype='int64')
print('================')
print('Confusion Matrix')
print('================')
CM = confusion_matrix(Y, y_pred)
print('True negatives: ',  CM[0,0])
print('False negatives: ', CM[1,0])
print('False positives: ', CM[0,1])
print('True positives: ',  CM[1,1])
```

```
================
Confusion Matrix
================
True negatives:  121
False negatives:  5
False positives:  3
True positives:  103
```

In [49]:
```
# Plot Loss and Accuracy by Epoch
plt.plot(history.history['accuracy'])
plt.plot(history.history['loss'])
plt.title('50 Epochs with LR= 0.001 and 10 layers NN')
plt.ylabel('Loss and Accuracy')
plt.xlabel('Epoch')
plt.legend(['accuracy', 'loss'], loc='center right')
plt.text(15,-.2, 'Results in 96.55% Accuracy and 14.86% Loss', ha='cente
r', va='top')
plt.show()
```



In [ ]: