Author: Daniel Lagos

https://github.com/danlagos/ANA500-Week-4

Date: 23 October 2022

ANA 500

Summary

Purpose: Test the hypothesis that there is no association between Depression and poverty after adjusting for general health and smoking habits in CHIS 2020.

Neural Network (NN) will be built to determine association. A decision tree was previously built for the same research question. Though this is not strictly a comparision of the results of decision tree vs this nueral network, they are relevant as they will assisst in framing our conclusions. It does not make sense to ignore previous work conducted.

Results from decision tree: Accuracy about 75%, recall score=0, and 56.6338% precision.

Results from NN: loss: 0.5388 - accuracy: 0.7526 - val_loss: 0.5275 - val_accuracy: 0.7635. These results are found below.

In [2]:
```
# import libraries
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
import pydot
import glob
import zipfile
```

In [3]:
```
%matplotlib inline
```

Bring in CHIS 2020 from local hardrive

take zipped file "adult.zip" Unzip. put file path in FILE_LOC

In [4]:
```
FILE_LOC = 'C:/Users/lagos/OneDrive/Documents/National University/8 - ANA 500/Week 1/h
df = pd.read_sas(FILE_LOC)
```

In [5]:
```
df.head()
```

Out[5]:

	AA5C	AA5G	AH37	AH44	AB1	AB17	AB40	AB41	AB18	AB43	...	RAKEDW71	RAKEDW72
0	-1.0	-1.0	-1.0	-1.0	3.0	2.0	-1.0	-1.0	-1.0	-1.0	...	1605.912814	1577.479222
1	-1.0	-1.0	-1.0	1.0	2.0	2.0	-1.0	-1.0	-1.0	-1.0	...	1558.197812	1513.430922
2	-1.0	-1.0	-1.0	-1.0	2.0	2.0	-1.0	-1.0	-1.0	-1.0	...	1211.750536	1212.772970
3	-1.0	-1.0	-1.0	-1.0	3.0	1.0	1.0	2.0	2.0	2.0	...	590.153646	604.817107
4	-1.0	1.0	-1.0	1.0	4.0	2.0	-1.0	-1.0	-1.0	-1.0	...	194.059102	188.468808

5 rows × 604 columns

In [6]:
```
df.shape
```

Out[6]:
```
(21949, 604)
```

In [7]:
```
df.describe()
```

Out[7]:

	AA5C	AA5G	AH37	AH44	AB1	AB17	A
count	21949.000000	21949.000000	21949.000000	21949.000000	21949.000000	21949.000000	21949.00(
mean	-0.960636	-0.552098	-0.326302	0.222379	2.341291	1.833888	-0.59
std	0.659912	1.168150	1.184560	0.991387	0.981429	0.372190	0.91
min	-9.000000	-9.000000	-1.000000	-1.000000	1.000000	1.000000	-1.00(
25%	-1.000000	-1.000000	-1.000000	-1.000000	2.000000	2.000000	-1.00(
50%	-1.000000	-1.000000	-1.000000	1.000000	2.000000	2.000000	-1.00(
75%	-1.000000	-1.000000	1.000000	1.000000	3.000000	2.000000	-1.00(
max	2.000000	2.000000	4.000000	2.000000	5.000000	2.000000	2.00(

8 rows × 603 columns

VAR DEPRESSION

In [8]:
```
df.AJ32.describe()
```

Out[8]:
```
count    21949.000000
mean         4.631783
std          0.734293
min         -2.000000
25%          5.000000
50%          5.000000
75%          5.000000
max          5.000000
Name: AJ32, dtype: float64
```

VAR SMOKING HABITS

In [9]:
```
df.SMOKING.describe()
```

Out[9]:
```
count    21949.000000
mean         2.638343
std          0.582503
min          1.000000
25%          2.000000
50%          3.000000
75%          3.000000
max          3.000000
Name: SMOKING, dtype: float64
```

VAR GENERAL HEALTH

In [10]:
```
df.AB1.describe()
```

Out[10]:
```
count    21949.000000
mean         2.341291
std          0.981429
min          1.000000
25%          2.000000
50%          2.000000
75%          3.000000
max          5.000000
Name: AB1, dtype: float64
```

POVERTY LEVELS AS LEVELS OF FEDERAL POVERTY LEVELS

In [11]:
```
df.POVLL.describe()
```

Out[11]:
```
count    21949.000000
mean         3.423983
std          0.973787
min          1.000000
25%          3.000000
50%          4.000000
75%          4.000000
max          4.000000
Name: POVLL, dtype: float64
```

In [12]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21949 entries, 0 to 21948
Columns: 604 entries, AA5C to RAKEDW80
dtypes: float64(603), object(1)
memory usage: 101.1+ MB
```

Filter data: Get rid of values of negative values

Also remove un-needed variables

In [13]:
```
df = df.loc[:, ['SMOKING', 'AB1', 'POVLL', 'AJ32']]
df = df[(df['AJ32'] > -1)]

smokingCurrent = df.SMOKING # CURRENT SMOKING HABITS - CAT
generalHealth = df.AB1 # GENERAL HEALTH CONDITION - CAT
povertyFPL = df.POVLL # FPL - CAT
depression = df.AJ32 # feeling depressed in last 30 days - CAT - DEP VAR
```

Confirm negative values are gone.

In [14]:
```
df.describe()
```

Out[14]:

	SMOKING	AB1	POVLL	AJ32
count	21944.000000	21944.000000	21944.000000	21944.000000
mean	2.638307	2.341232	3.423943	4.633294
std	0.582533	0.981413	0.973866	0.727520
min	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000	5.000000
50%	3.000000	2.000000	4.000000	5.000000
75%	3.000000	3.000000	4.000000	5.000000
max	3.000000	5.000000	4.000000	5.000000

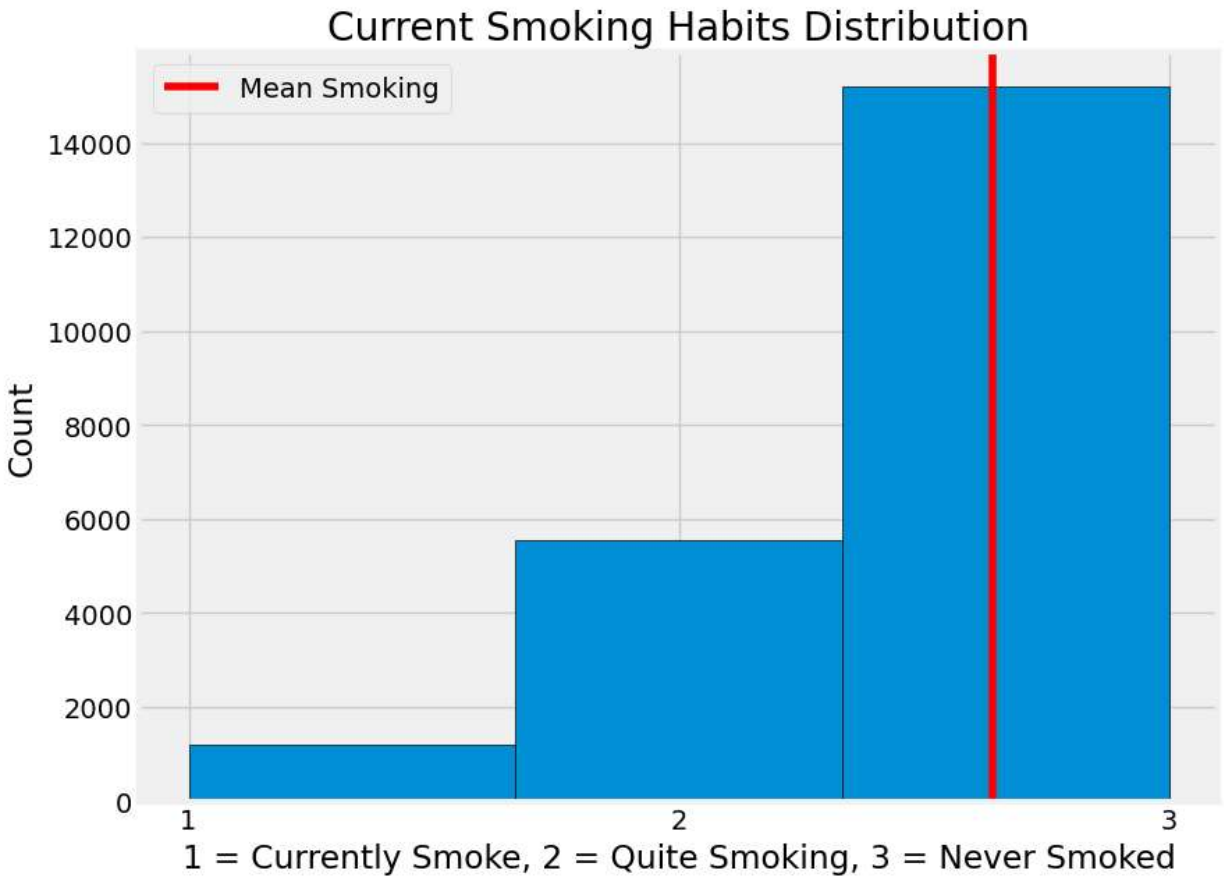
Histograms

Current smoking habits

In [15]:
```
smoking_mean=2.638307
smoking_color='red'
plt.figure(figsize=(10,7))
plt.style.use('fivethirtyeight')
plt.hist(df['SMOKING'], bins=3, edgecolor='black')
plt.title("Current Smoking Habits Distribution")
plt.xlabel('1 = Currently Smoke, 2 = Quite Smoking, 3 = Never Smoked')
plt.xticks([1, 2, 3])
```

```
plt.ylabel('Count')
plt.axvline(smoking_mean, color=smoking_color, label='Mean Smoking')
plt.legend()
```

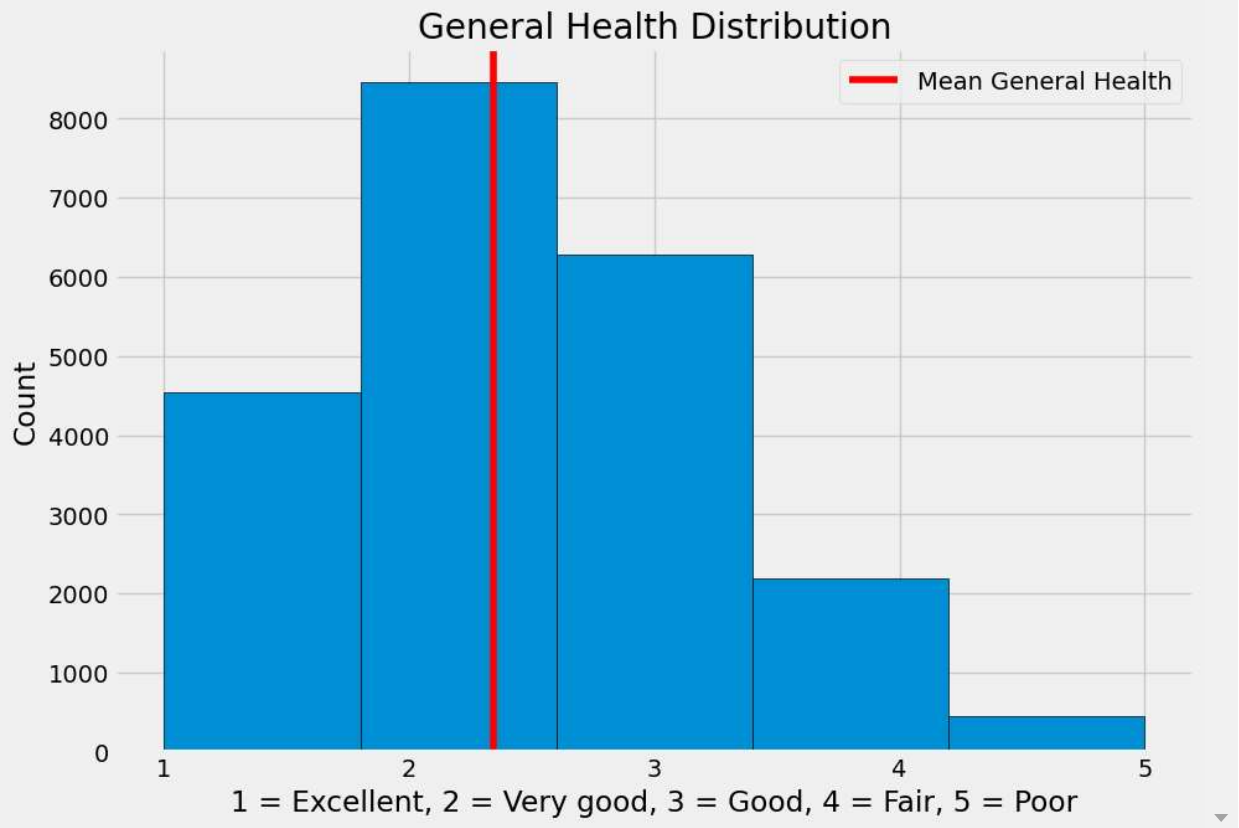
Out[15]: <matplotlib.legend.Legend at 0x1d978f28580>



General Health

In [16]:
```
#g_h_labels = ['1:excellent', '2:very good', '3: good', '4: fair', '5: poor']
gen_health_mean=2.341232
gen_health_color='red'
plt.figure(figsize=(10,7))
plt.style.use('fivethirtyeight')
plt.hist(df['AB1'], bins=5, edgecolor='black')
plt.title("General Health Distribution")
plt.xlabel('1 = Excellent, 2 = Very good, 3 = Good, 4 = Fair, 5 = Poor')
plt.xticks([1, 2, 3, 4, 5])
plt.ylabel('Count')
plt.axvline(gen_health_mean, color=gen_health_color, label='Mean General Health')
plt.legend()
```

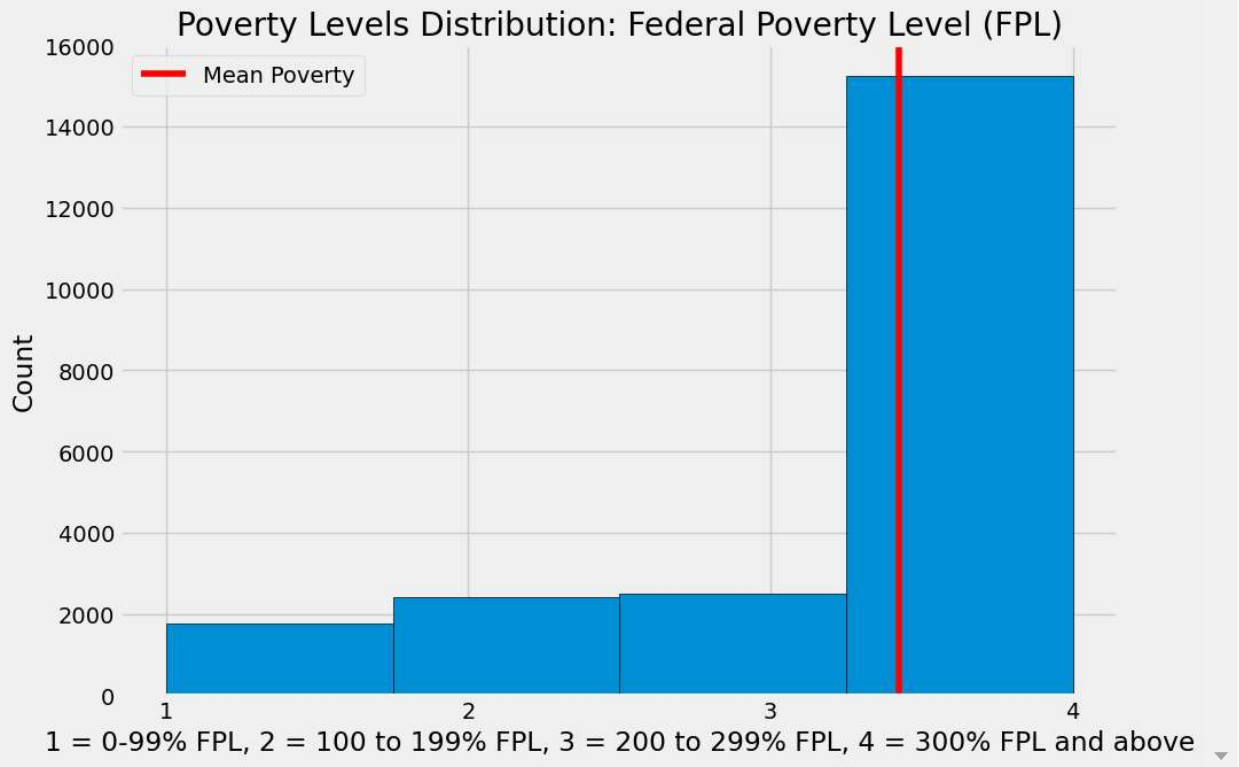
Out[16]: <matplotlib.legend.Legend at 0x1d977f4ec10>



Poverty level, as percentage of federal poverty level (FPL)

In [17]:
```
poverty_mean=3.423943
poverty_color='red'
plt.figure(figsize=(10,7))
plt.style.use('fivethirtyeight')
plt.hist(df['POVLL'], bins=4, edgecolor='black')
plt.title("Poverty Levels Distribution: Federal Poverty Level (FPL)")
plt.xlabel('1 = 0-99% FPL, 2 = 100 to 199% FPL, 3 = 200 to 299% FPL, 4 = 300% FPL and
plt.xticks([1, 2, 3, 4])
plt.ylabel('Count')
plt.axvline(poverty_mean, color=poverty_color, label='Mean Poverty')
plt.legend()
```

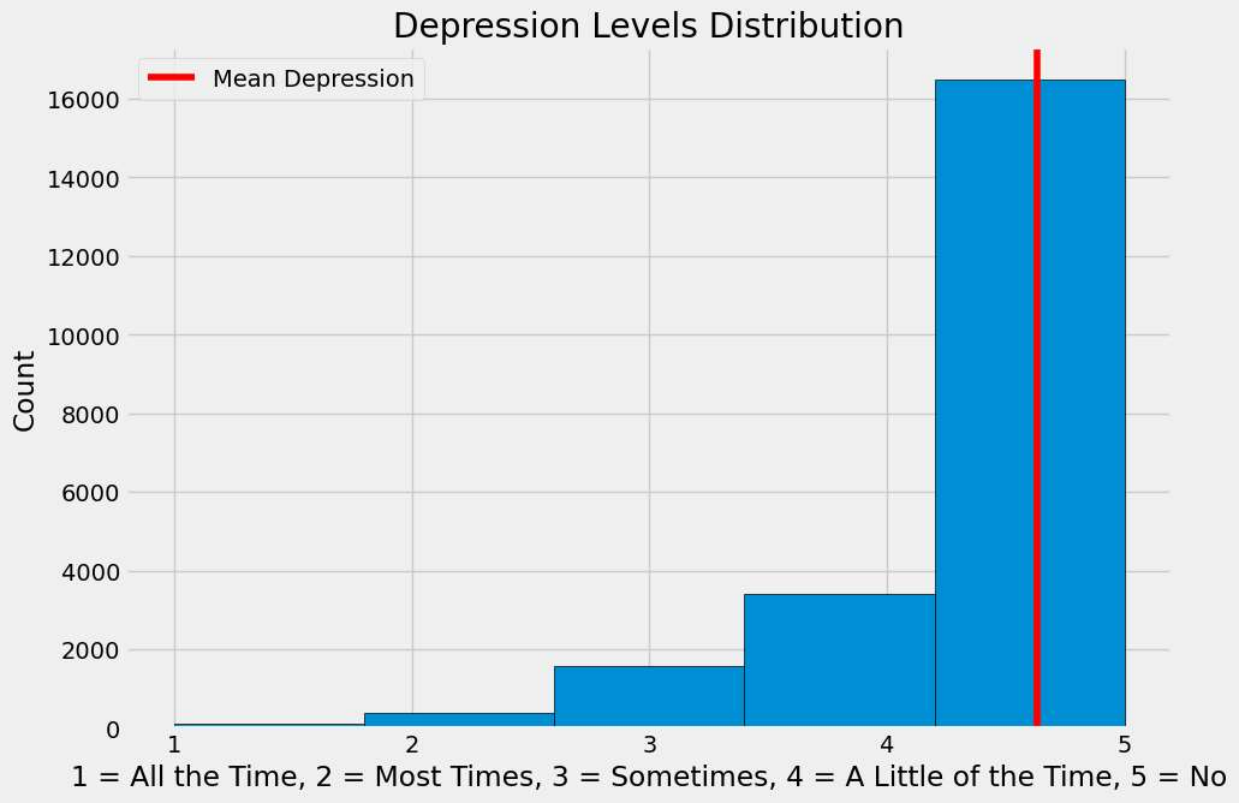
Out[17]:
```
<matplotlib.legend.Legend at 0x1d967f9c190>
```



Depression

In [18]:
```
depression_mean=4.633294
depression_color='red'
plt.figure(figsize=(10,7))
plt.style.use('fivethirtyeight')
plt.hist(df['AJ32'], bins=5, edgecolor='black')
plt.title("Depression Levels Distribution")
plt.xlabel('1 = All the Time, 2 = Most Times, 3 = Sometimes, 4 = A Little of the Time,
plt.xticks([1, 2, 3, 4, 5])
plt.ylabel('Count')
plt.axvline(depression_mean, color=depression_color, label='Mean Depression')
plt.legend()
```

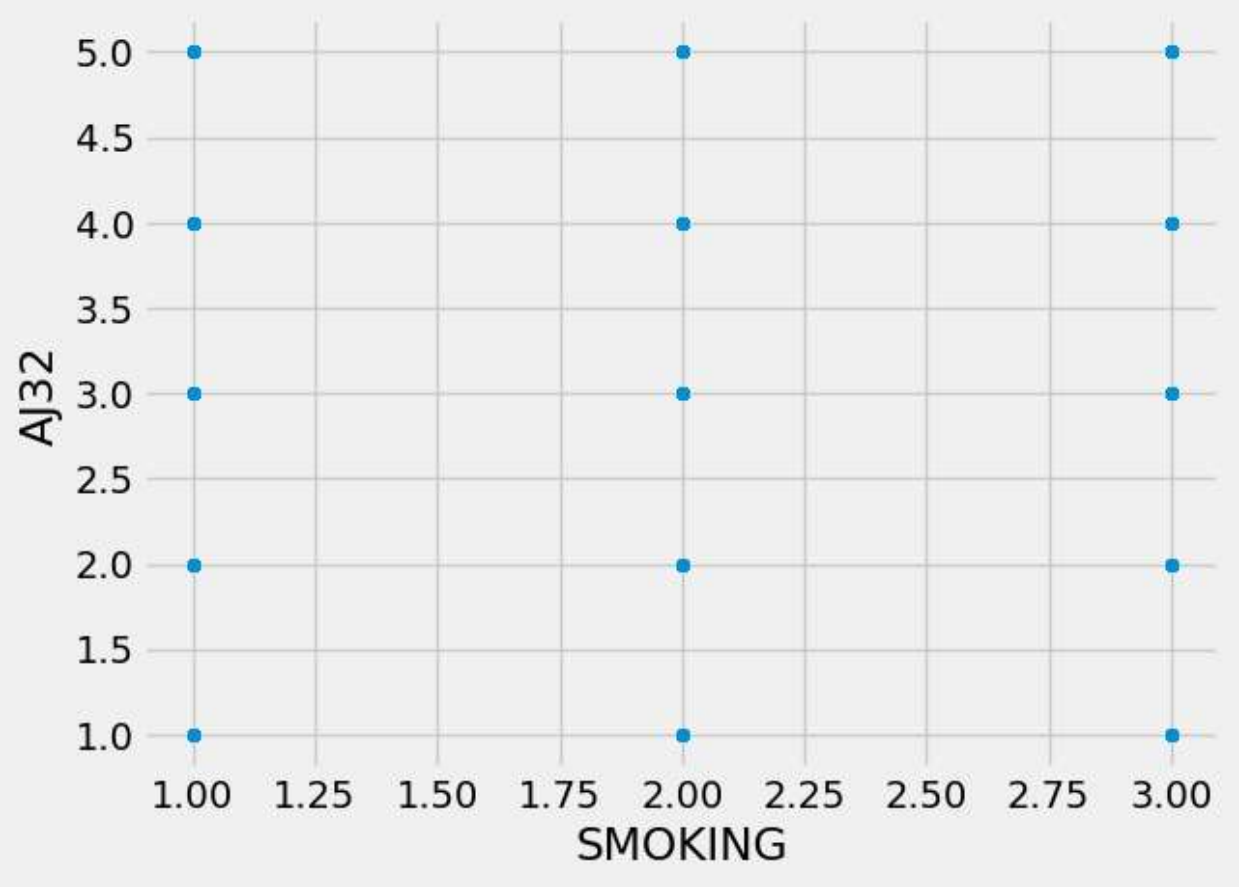
Out[18]:
```
<matplotlib.legend.Legend at 0x1d901397b50>
```



Smoking vs Depression

In [19]:
```
df.plot.scatter(x='SMOKING', y='AJ32')
```

Out[19]:
```
<AxesSubplot:xlabel='SMOKING', ylabel='AJ32'>
```



Box plots

Boxplot: Depression

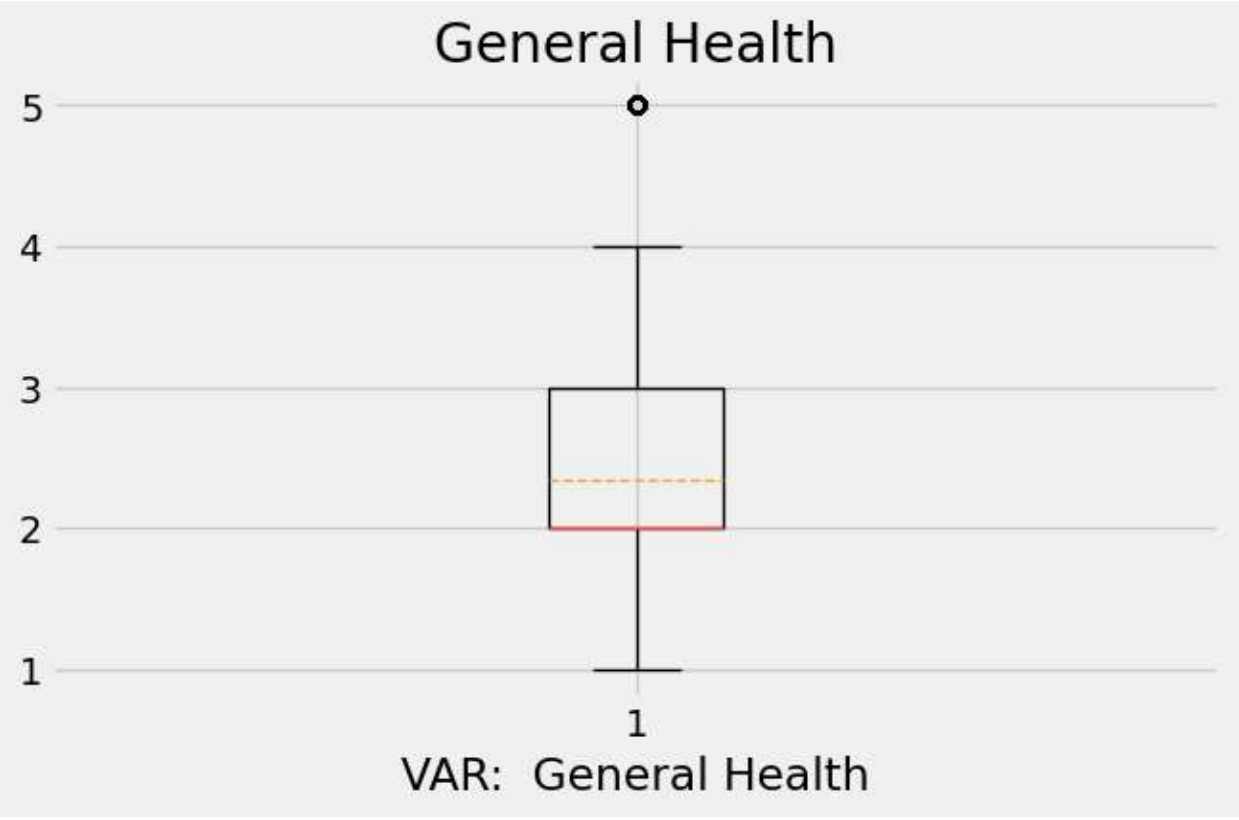
In [20]:
```
plt.figure(figsize=(8,4))
plt.boxplot(df['AJ32'], showmeans=True, meanline=True)
plt.title("Depression")
plt.xlabel('VAR:  Depression')
```

Out[20]:
```
Text(0.5, 0, 'VAR:  Depression')
```

Boxplot: General Health

In [21]:
```
plt.figure(figsize=(7,4))
plt.boxplot(df['AB1'], showmeans=True, meanline=True)
plt.title("General Health")
plt.xlabel('VAR:  General Health')
```

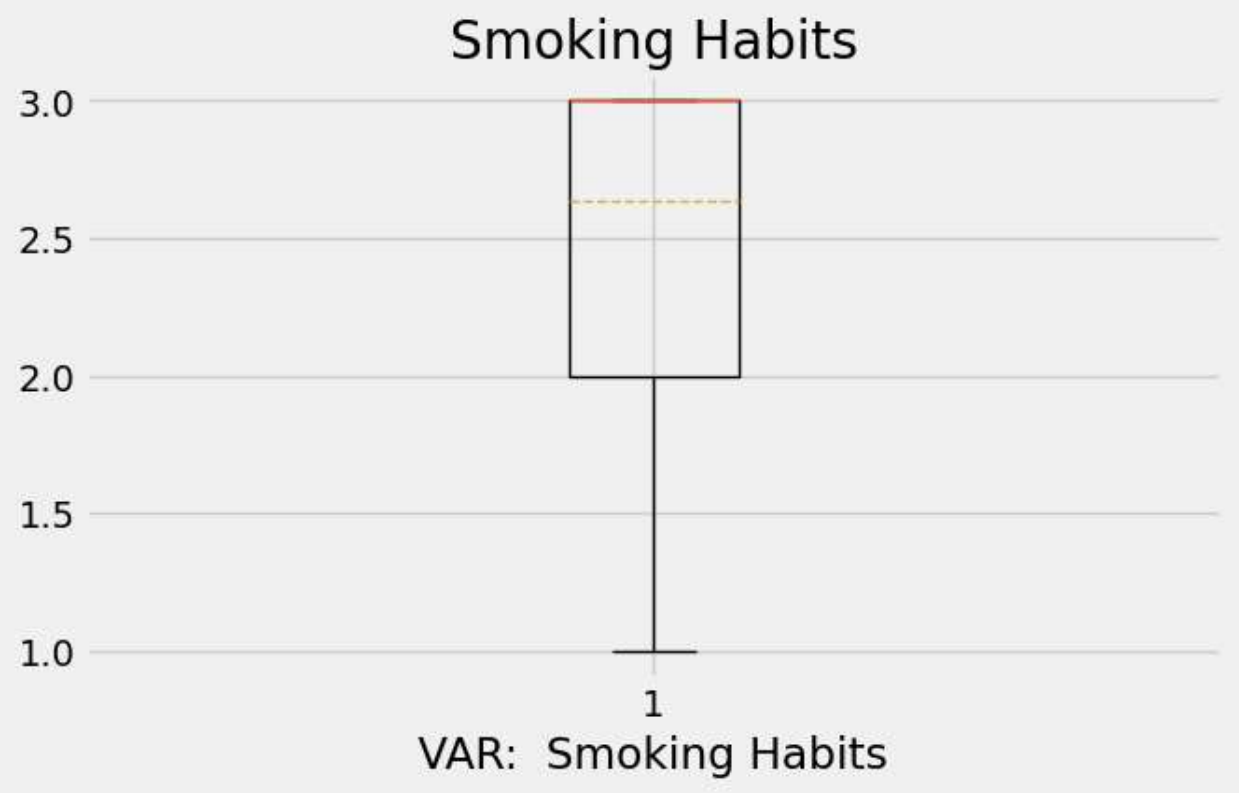
Out[21]:
```
Text(0.5, 0, 'VAR:  General Health')
```



Boxplot: Smoking Habits

In [22]:
```
plt.figure(figsize=(7,4))
plt.boxplot(df['SMOKING'], showmeans=True, meanline=True)
plt.title("Smoking Habits")
plt.xlabel('VAR:  Smoking Habits')
```

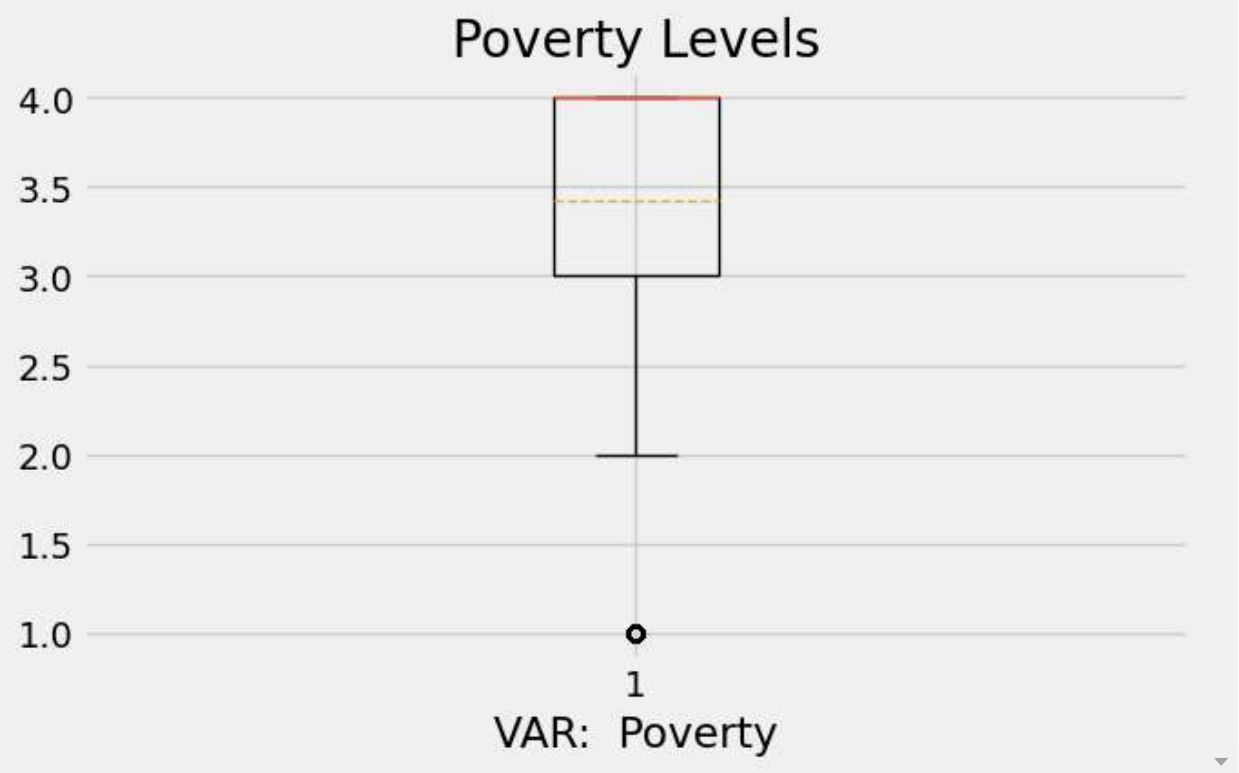
Out[22]:
```
Text(0.5, 0, 'VAR:  Smoking Habits')
```



Boxplot: Poverty

In [23]:
```
plt.figure(figsize=(7,4))
plt.boxplot(df['POVLL'], showmeans=True, meanline=True)
plt.title("Poverty Levels")
plt.xlabel('VAR:  Poverty')
```

Out[23]:
```
Text(0.5, 0, 'VAR:  Poverty')
```



Correlation Matrix

In [24]:
```
df.corr()
```

Out[24]:

	SMOKING	AB1	POVLL	AJ32
SMOKING	1.000000	-0.145927	0.059351	0.037362
AB1	-0.145927	1.000000	-0.211444	-0.197935
POVLL	0.059351	-0.211444	1.000000	0.135364
AJ32	0.037362	-0.197935	0.135364	1.000000

Rename Variables

Create new data frame with clean data, rename variables.

In [25]:
```
df_renamed = df.rename(columns = {'AB1': "GEN_HEALTH", "POVLL": "POVERTY", 'AJ32': 'DE
```

In [26]:
```
df_renamed.describe()
```

Out[26]:

	SMOKING	GEN_HEALTH	POVERTY	DEPRESSION
count	21944.000000	21944.000000	21944.000000	21944.000000
mean	2.638307	2.341232	3.423943	4.633294
std	0.582533	0.981413	0.973866	0.727520
min	1.000000	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000	5.000000
50%	3.000000	2.000000	4.000000	5.000000
75%	3.000000	3.000000	4.000000	5.000000
max	3.000000	5.000000	4.000000	5.000000

Summary

DEPRESSION: Dependent variable. Coded in CHIS as AJ32. Survey question: Feeling depressed in the past 30 days?

Coding:

```
1 = all the time
2 = most of the time
3 = some of the time
4 = a little of the time
5 = not at all
```

POVERTY: Coded as POVLL in CHIS. Calculated from respondants annual income and converted to reflect federal poverty levels (PFL).

Coding:

```
1 = 0-99% FPL
2 = 100 to 199% FPL
3 = 200 to 299% FPL
4 = 300% FPL and above
```

GEN_HEALTH: Coded in CHIS as AB1. Tracks respondants self reported general health condition.

Coding:

```
1 = excellent
2 = very good
3 = good
4 = fair
```

```
5 = poor
```

SMOKING: Coded in CHIS as SMOKING. Tracks respondants self reported smoking habits. If resondant indicated not smoking >= 100 cigarettes in lifetime then categorized as "never smoked regularly."

```
Coding

    1 = currently smokes
    2 = quit smoking
    3 = never smoked regularly
```

Re-create model with Var DEPRESSION as binary

make Var Depression Binary.

If DEPRESSION = 5 then newDep = 2 Meaning, not-depressed

if DEPRESSION <= 5 then newDep = 1 Meaning depressed.

In [27]:
```
df_renamed.loc[df_renamed['DEPRESSION'] > 4, 'NEW_DEP'] = 1 # NOT DEPRESSED
df_renamed.loc[df_renamed['DEPRESSION'] <= 4, 'NEW_DEP'] = 0 # DEPRESSED
df_renamed.describe()
```

Out[27]:

	SMOKING	GEN_HEALTH	POVERTY	DEPRESSION	NEW_DEP
count	21944.000000	21944.000000	21944.000000	21944.000000	21944.000000
mean	2.638307	2.341232	3.423943	4.633294	0.751868
std	0.582533	0.981413	0.973866	0.727520	0.431938
min	1.000000	1.000000	1.000000	1.000000	0.000000
25%	2.000000	2.000000	3.000000	5.000000	1.000000
50%	3.000000	2.000000	4.000000	5.000000	1.000000
75%	3.000000	3.000000	4.000000	5.000000	1.000000
max	3.000000	5.000000	4.000000	5.000000	1.000000

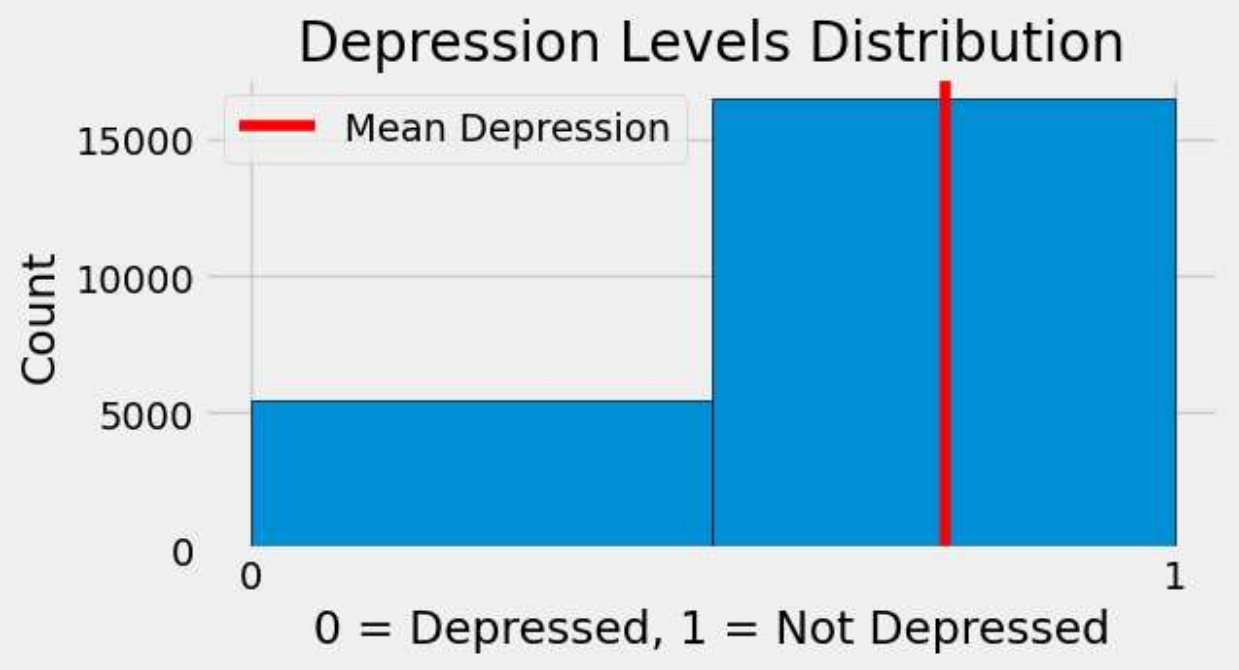
Visualiztion for new variable of DEPRESSION

Histogram: NEW_DEP

In [47]:
```
depression_mean=0.751868
depression_color='red'
plt.figure(figsize=(6,3))
```

```
plt.style.use('fivethirtyeight')
plt.hist(df_renamed['NEW_DEP'], bins=2, edgecolor='black')
plt.title("Depression Levels Distribution")
plt.xlabel('0 = Depressed, 1 = Not Depressed')
plt.xticks([0, 1])
plt.ylabel('Count')
plt.axvline(depression_mean, color=depression_color, label='Mean Depression')
plt.legend()
```

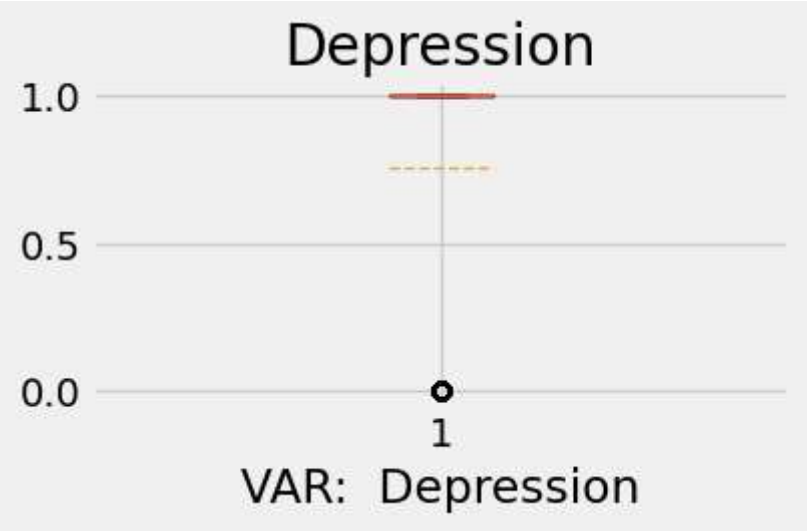
Out[47]:
```
<matplotlib.legend.Legend at 0x1d901f05b80>
```



Box plot: NEW_DEP

In [48]:
```
plt.figure(figsize=(4,2))
plt.boxplot(df_renamed['NEW_DEP'], showmeans=True, meanline=True)
plt.title("Depression")
plt.xlabel('VAR:  Depression')
```

Out[48]:
```
Text(0.5, 0, 'VAR:  Depression')
```



Run Neural Network with new variable for Depression

Create "Y_New" dataframe out of Depression

In [30]:
```
Y_New = df_renamed[['NEW_DEP']]
print(type(Y_New))
Y_New.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
```

Out[30]:

	NEW_DEP
count	21944.000000
mean	0.751868
std	0.431938
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

Create "X_New" dataframe out of POVERTY, GEN_HEALTH, SMOKING

In [31]:
```
ind_var_feat = df_renamed.columns[0:-2]
ind_var_feat
```

Out[31]:
```
Index(['SMOKING', 'GEN_HEALTH', 'POVERTY'], dtype='object')
```

In [32]:
```
X_New = df_renamed[ind_var_feat]
X_New.describe()
```

Out[32]:

	SMOKING	GEN_HEALTH	POVERTY
count	21944.000000	21944.000000	21944.000000
mean	2.638307	2.341232	3.423943
std	0.582533	0.981413	0.973866
min	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000
50%	3.000000	2.000000	4.000000
75%	3.000000	3.000000	4.000000
max	3.000000	5.000000	4.000000

Split data set into training and testing sets

In [33]:
```
X_New_train, X_New_test, Y_New_train, Y_New_test = train_test_split(X_New, Y_New,
                                                                    test_size=0.4,
                                                                    random_state=42
                                                                   )
X_New_train, X_val, Y_New_train, Y_val = train_test_split(X_New_train,
                                                          Y_New_train,
                                                         test_size=.15,
                                                         random_state=42)
```

confirm splits

In [34]:
```
print(type(X_New_train))
print(type(X_New_test))
print(type(X_val))
print(type(Y_New_train))
print(type(Y_New_test))
print(type(Y_val))
```

```
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
<class 'pandas.core.frame.DataFrame'>
```

In [35]:
```
#  Look at new data sets for confirmation
X_New_train.describe()
```

Out[35]:

	SMOKING	GEN_HEALTH	POVERTY
count	11191.000000	11191.000000	11191.000000
mean	2.640961	2.342597	3.428737
std	0.579620	0.979070	0.969458
min	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000
50%	3.000000	2.000000	4.000000
75%	3.000000	3.000000	4.000000
max	3.000000	5.000000	4.000000

In [36]:
```
X_New_test.describe()
```

Out[36]:

	SMOKING	GEN_HEALTH	POVERTY
count	8778.000000	8778.000000	8778.000000
mean	2.637503	2.338004	3.411825
std	0.583520	0.979460	0.982370
min	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000
50%	3.000000	2.000000	4.000000
75%	3.000000	3.000000	4.000000
max	3.000000	5.000000	4.000000

In [37]:
```
X_val.describe()
```

Out[37]:

	SMOKING	GEN_HEALTH	POVERTY
count	1975.000000	1975.000000	1975.000000
mean	2.626835	2.347848	3.450633
std	0.594631	1.003547	0.960448
min	1.000000	1.000000	1.000000
25%	2.000000	2.000000	3.000000
50%	3.000000	2.000000	4.000000
75%	3.000000	3.000000	4.000000
max	3.000000	5.000000	4.000000

In [38]:
```
Y_New_test.describe()
```

Out[38]:

	NEW_DEP
count	8778.000000
mean	0.749829
std	0.433136
min	0.000000
25%	0.000000
50%	1.000000
75%	1.000000
max	1.000000

In [39]:
```
Y_New_train.describe()
```

Out[39]:

	NEW_DEP
count	11191.000000
mean	0.751765
std	0.432008
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

In [50]:
```
Y_val.describe()
```

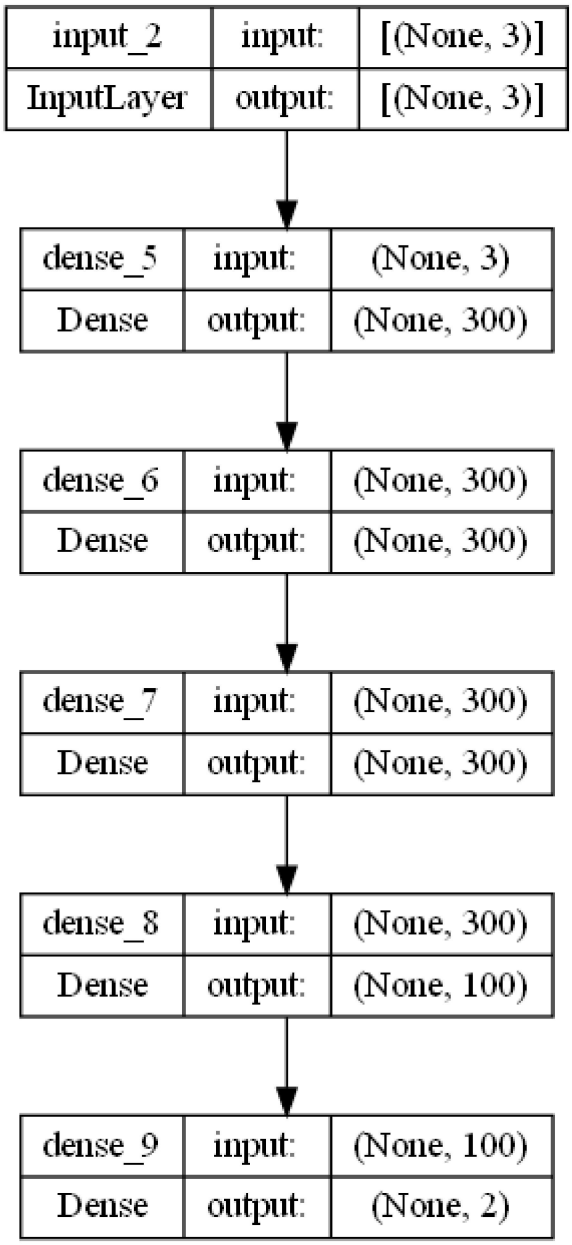
Out[50]:

	NEW_DEP
count	1975.000000
mean	0.761519
std	0.426263
min	0.000000
25%	1.000000
50%	1.000000
75%	1.000000
max	1.000000

Build to model

In [51]:
```
input_shape=3
num_classes=2
model = keras.models.Sequential()
model.add(keras.Input(shape=input_shape, dtype='float64'))
model.add(keras.layers.Dense(300, activation='relu'))
model.add(keras.layers.Dense(300, activation='relu'))
model.add(keras.layers.Dense(300, activation='relu'))
model.add(keras.layers.Dense(100, activation='relu'))
model.add(keras.layers.Dense(num_classes, activation='softmax'))

# nn_model = make_model(input_shape=3, num_classes=2)
keras.utils.plot_model(model, show_shapes=True)
```

Out[51]:



In [52]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 300)               1200

 dense_6 (Dense)             (None, 300)               90300

 dense_7 (Dense)             (None, 300)               90300

 dense_8 (Dense)             (None, 100)               30100

 dense_9 (Dense)             (None, 2)                 202

=================================================================
Total params: 212,102
Trainable params: 212,102
Non-trainable params: 0
_________________________________________________________________
```

In [53]:
```
model.layers
```

Out[53]:
```
[<keras.layers.core.dense.Dense at 0x1d90cba6520>,
 <keras.layers.core.dense.Dense at 0x1d9013380d0>,
 <keras.layers.core.dense.Dense at 0x1d95be906a0>,
 <keras.layers.core.dense.Dense at 0x1d9000f78e0>,
 <keras.layers.core.dense.Dense at 0x1d90cba63a0>]
```

Compile the model

In [54]:
```
model.compile(loss='sparse_categorical_crossentropy', optimizer="sgd", metrics=['accur
```

Fit the model

In [55]:
```
history = model.fit(X_New_train, Y_New_train, epochs=50, validation_data= (X_val,Y_val
```

```
Epoch 1/50
350/350 [==============================] - 5s 6ms/step - loss: 0.5471 - accuracy: 0.7
524 - val_loss: 0.5416 - val_accuracy: 0.7595
Epoch 2/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5434 - accuracy: 0.7
519 - val_loss: 0.5317 - val_accuracy: 0.7630
Epoch 3/50
350/350 [==============================] - 3s 7ms/step - loss: 0.5429 - accuracy: 0.7
524 - val_loss: 0.5321 - val_accuracy: 0.7635
Epoch 4/50
350/350 [==============================] - 3s 8ms/step - loss: 0.5424 - accuracy: 0.7
518 - val_loss: 0.5307 - val_accuracy: 0.7630
Epoch 5/50
350/350 [==============================] - 3s 8ms/step - loss: 0.5419 - accuracy: 0.7
520 - val_loss: 0.5305 - val_accuracy: 0.7630
Epoch 6/50
350/350 [==============================] - 2s 7ms/step - loss: 0.5417 - accuracy: 0.7
505 - val_loss: 0.5311 - val_accuracy: 0.7625
Epoch 7/50
350/350 [==============================] - 3s 10ms/step - loss: 0.5414 - accuracy: 0.
7520 - val_loss: 0.5301 - val_accuracy: 0.7635
Epoch 8/50
350/350 [==============================] - 2s 7ms/step - loss: 0.5411 - accuracy: 0.7
514 - val_loss: 0.5305 - val_accuracy: 0.7625
Epoch 9/50
350/350 [==============================] - 3s 9ms/step - loss: 0.5403 - accuracy: 0.7
527 - val_loss: 0.5293 - val_accuracy: 0.7646
Epoch 10/50
350/350 [==============================] - 1s 4ms/step - loss: 0.5407 - accuracy: 0.7
520 - val_loss: 0.5296 - val_accuracy: 0.7620
Epoch 11/50
350/350 [==============================] - 1s 4ms/step - loss: 0.5405 - accuracy: 0.7
527 - val_loss: 0.5297 - val_accuracy: 0.7630
Epoch 12/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5403 - accuracy: 0.7
530 - val_loss: 0.5292 - val_accuracy: 0.7635
Epoch 13/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5401 - accuracy: 0.7
520 - val_loss: 0.5299 - val_accuracy: 0.7620
Epoch 14/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5402 - accuracy: 0.7
531 - val_loss: 0.5295 - val_accuracy: 0.7635
Epoch 15/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5400 - accuracy: 0.7
536 - val_loss: 0.5322 - val_accuracy: 0.7600
Epoch 16/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5401 - accuracy: 0.7
519 - val_loss: 0.5287 - val_accuracy: 0.7635
Epoch 17/50
350/350 [==============================] - 2s 4ms/step - loss: 0.5394 - accuracy: 0.7
533 - val_loss: 0.5297 - val_accuracy: 0.7635
Epoch 18/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5398 - accuracy: 0.7
526 - val_loss: 0.5295 - val_accuracy: 0.7620
Epoch 19/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5398 - accuracy: 0.7
527 - val_loss: 0.5299 - val_accuracy: 0.7620
Epoch 20/50
350/350 [==============================] - 2s 7ms/step - loss: 0.5397 - accuracy: 0.7
518 - val_loss: 0.5288 - val_accuracy: 0.7635
```

```
Epoch 21/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5397 - accuracy: 0.7
522 - val_loss: 0.5290 - val_accuracy: 0.7635
Epoch 22/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5397 - accuracy: 0.7
521 - val_loss: 0.5295 - val_accuracy: 0.7635
Epoch 23/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5395 - accuracy: 0.7
533 - val_loss: 0.5327 - val_accuracy: 0.7585
Epoch 24/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5396 - accuracy: 0.7
531 - val_loss: 0.5288 - val_accuracy: 0.7605
Epoch 25/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5395 - accuracy: 0.7
525 - val_loss: 0.5289 - val_accuracy: 0.7641
Epoch 26/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5393 - accuracy: 0.7
521 - val_loss: 0.5289 - val_accuracy: 0.7595
Epoch 27/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5395 - accuracy: 0.7
523 - val_loss: 0.5287 - val_accuracy: 0.7635
Epoch 28/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5391 - accuracy: 0.7
512 - val_loss: 0.5284 - val_accuracy: 0.7635
Epoch 29/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5390 - accuracy: 0.7
522 - val_loss: 0.5291 - val_accuracy: 0.7635
Epoch 30/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5392 - accuracy: 0.7
521 - val_loss: 0.5286 - val_accuracy: 0.7595
Epoch 31/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5390 - accuracy: 0.7
517 - val_loss: 0.5299 - val_accuracy: 0.7595
Epoch 32/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5390 - accuracy: 0.7
526 - val_loss: 0.5283 - val_accuracy: 0.7595
Epoch 33/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5392 - accuracy: 0.7
523 - val_loss: 0.5284 - val_accuracy: 0.7605
Epoch 34/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5391 - accuracy: 0.7
518 - val_loss: 0.5288 - val_accuracy: 0.7635
Epoch 35/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5393 - accuracy: 0.7
523 - val_loss: 0.5275 - val_accuracy: 0.7635
Epoch 36/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5390 - accuracy: 0.7
515 - val_loss: 0.5278 - val_accuracy: 0.7635
Epoch 37/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5391 - accuracy: 0.7
519 - val_loss: 0.5277 - val_accuracy: 0.7635
Epoch 38/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5393 - accuracy: 0.7
517 - val_loss: 0.5288 - val_accuracy: 0.7595
Epoch 39/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5389 - accuracy: 0.7
527 - val_loss: 0.5296 - val_accuracy: 0.7585
Epoch 40/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5390 - accuracy: 0.7
517 - val_loss: 0.5277 - val_accuracy: 0.7635
```

```
Epoch 41/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5389 - accuracy: 0.7
528 - val_loss: 0.5294 - val_accuracy: 0.7585
Epoch 42/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5387 - accuracy: 0.7
530 - val_loss: 0.5289 - val_accuracy: 0.7635
Epoch 43/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5387 - accuracy: 0.7
527 - val_loss: 0.5278 - val_accuracy: 0.7641
Epoch 44/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5387 - accuracy: 0.7
525 - val_loss: 0.5277 - val_accuracy: 0.7635
Epoch 45/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5385 - accuracy: 0.7
539 - val_loss: 0.5284 - val_accuracy: 0.7585
Epoch 46/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5385 - accuracy: 0.7
522 - val_loss: 0.5284 - val_accuracy: 0.7630
Epoch 47/50
350/350 [==============================] - 2s 6ms/step - loss: 0.5385 - accuracy: 0.7
522 - val_loss: 0.5285 - val_accuracy: 0.7595
Epoch 48/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5385 - accuracy: 0.7
518 - val_loss: 0.5277 - val_accuracy: 0.7635
Epoch 49/50
350/350 [==============================] - 2s 5ms/step - loss: 0.5386 - accuracy: 0.7
519 - val_loss: 0.5274 - val_accuracy: 0.7635
Epoch 50/50
350/350 [==============================] - 1s 4ms/step - loss: 0.5385 - accuracy: 0.7
524 - val_loss: 0.5275 - val_accuracy: 0.7595
```

Initial Pass: The model is not learning. Something is wrong, I want to investigate, but my initial thoughts are that a neural network is not the right algorithm for such a simple problem (there are a total of 4 variables).

```
Actions taken on initial pass:  changed coding to depression 0 =
depressed, 1 = not depressed

Results:  model is now learning, but at a low rate.  I am running 50
epochs, I think I need to add more layers.  This will be my next
course of action.  loss: 0.5390 - accuracy: 0.7528 - val_loss: 0.5292
- val_accuracy: 0.7641.
```

Second pass: not much improvement.

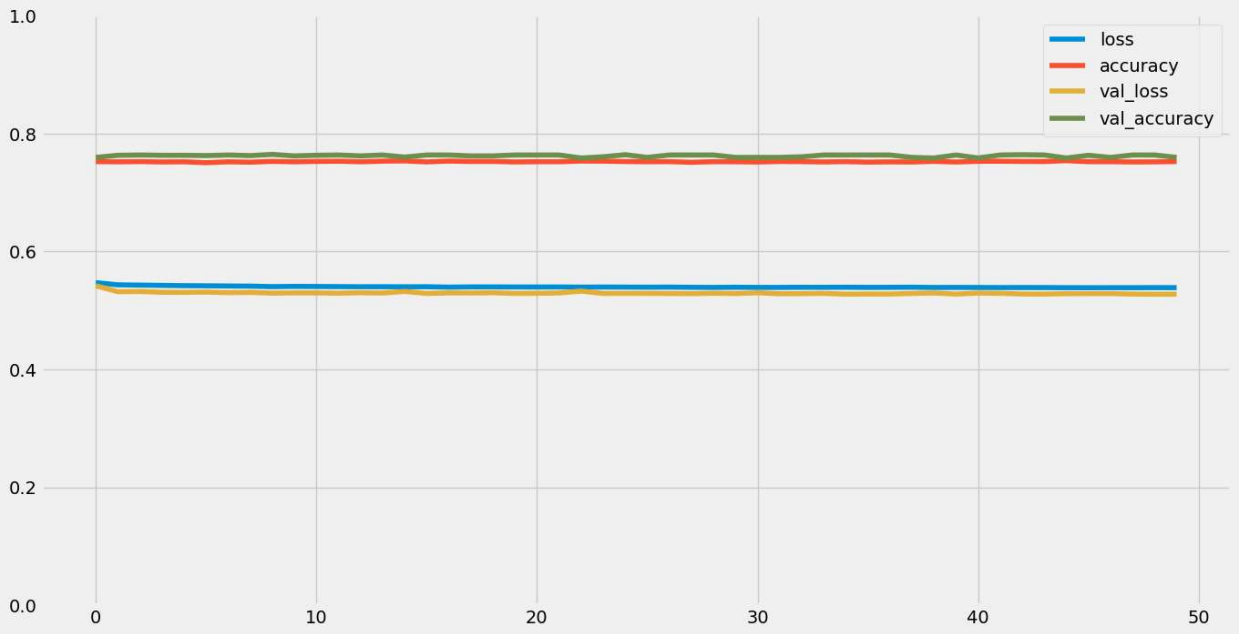
```
actions taken at second pass:  added two more hidden layers with 300
neurons each.  Ket same activation function.

Results:  loss: 0.5388 - accuracy: 0.7526 - val_loss: 0.5275 -
val_accuracy: 0.7635, essentially the same numbers
```

Notes, results may be slightly different as model may have been re-run after second pass.

Model Performance

In [56]:
```
pd.DataFrame(history.history).plot(figsize=(15,8))
plt.grid(True)
plt.gca().set_ylim(0,1)
```

Out[56]:
```
(0.0, 1.0)
```



intial pass: We have 0 or nan values for all metrics. After changes results are better, but the metrics look like horizontal lines.

second pass: I still have two sets of horizontal lines, they have not moved from their location relative to the y-axis

Evaluate model on test set

In [57]:
```
model.evaluate(X_New_test, Y_New_test)
```

```
275/275 [==============================] - 1s 3ms/step - loss: 0.5462 - accuracy: 0.7
478
```

Out[57]:
```
[0.5462146401405334, 0.7477785348892212]
```

Initial pass: again, 0 or nan values for accuracy and loss. After chaning dependant var the following results were prodiced

```
loss: 0.5488 - accuracy: 0.7505
```

second pass: After adding two hidden layers the following reulsts were produced

```
loss: 0.5463 - accuracy: 0.7509
```

Predictions probability

In [58]:
```
y_prob = model.predict(X_New_test)
y_classes = y_prob.argmax(axis=-1)
y_classes
```

```
275/275 [==============================] - 1s 3ms/step
```
Out[58]:
```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

Initial pass after changing dep var: array([1, 1, 1, ..., 1, 1, 1], dtype=int64)

second pass: array([1, 1, 1, ..., 1, 1, 1], dtype=int64)

no significant changes

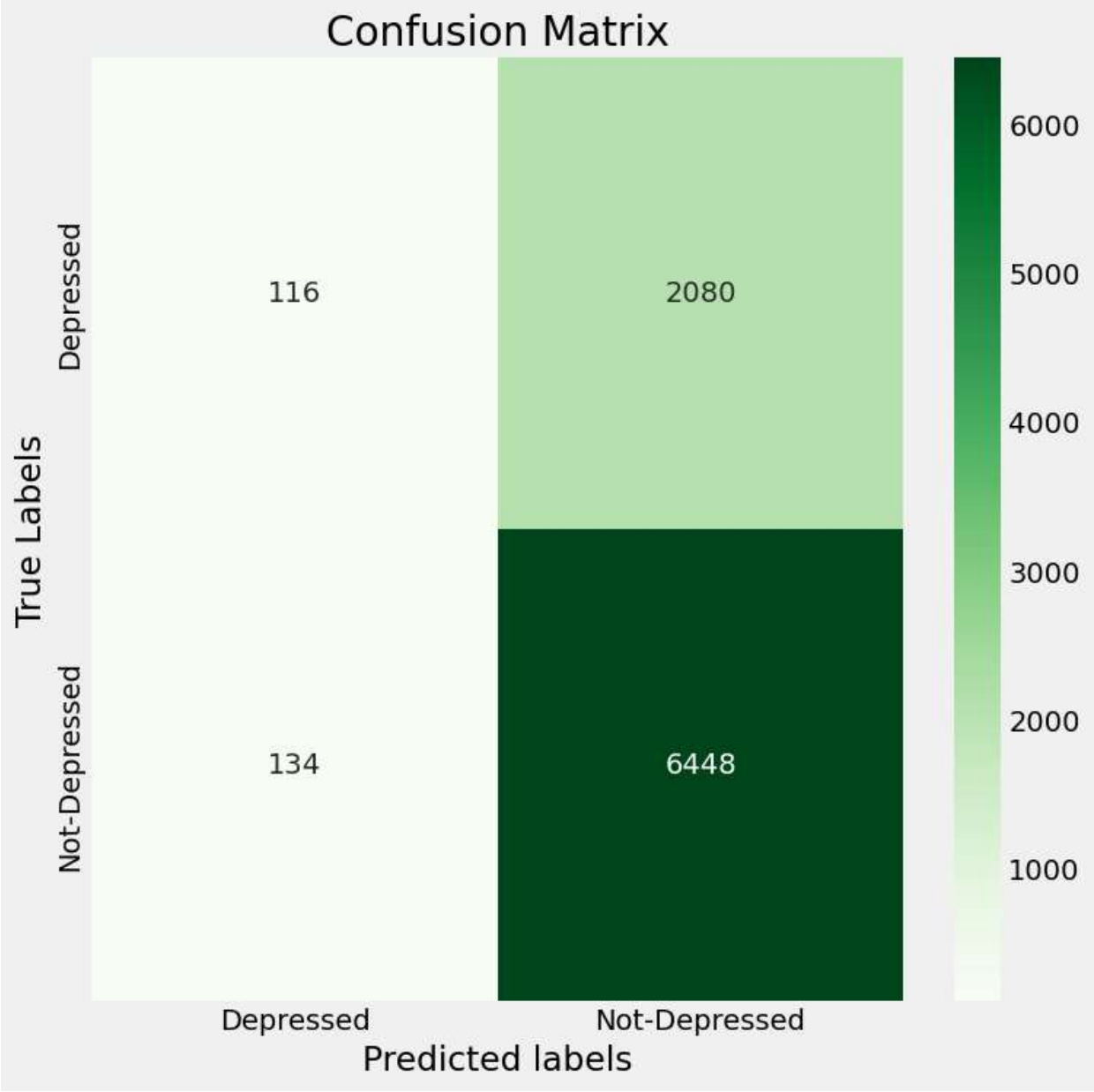
Confusion Matrix

In [59]:
```
confusion_matrix = tf.math.confusion_matrix(Y_New_test, y_classes)
confusion_matrix
```

Out[59]:
```
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=
array([[ 116, 2080],
       [ 134, 6448]])>
```

Create a better visualization of the confusion matrix

In [61]:
```
class_names = ['Depressed', 'Not-Depressed']

fig = sns.heatmap(confusion_matrix, annot=True, fmt="g", cmap="Greens")
fig.set_xlabel('Predicted labels')
fig.set_ylabel('True Labels')
fig.set_title("Confusion Matrix")
fig.xaxis.set_ticklabels(class_names)
fig.yaxis.set_ticklabels(class_names)
fig.figure.set_size_inches(8, 8)
```



initial pass: not getting results

second pass: appears that we have high misclassifications of not-depressed. We wnt to see diagonoal boxes I still beleive I should add two more hidden layers

Conclusions

loss: 0.5388 - accuracy: 0.7526 - val_loss: 0.5275 - val_accuracy: 0.7635

a lot of misclassification of "depressed' to "not depressed"

It must be noted that the neural network prduced similar accuracy scores as the decision tree from last week. Like last weeks model, the neural network cannot properly categorize "depressed" given the variables provided.

There was not much difference in accuracy/validation-accuracy when the number of hidden layers was increased.

The NN appears to have appropriate optimization and activation functions.

I do not beleive the model's hyper-parameters are the issue, I now beleive that the independant varaibles are not viable variables for predicting depression. Namely, they are not enough to adequatly predict DEPRESSION.

According to results from this neural network poverty, general health, and smoking habits are not strong predictors of depression. The three variables do predict "not-depressed" but has difficulty distinguishing "depressedd" and is misclassigying them as "depressed."

Reccomend two courses of action:

```
1.  Conduct a thorough test on neural networks hyper-parameters, to
confirm that changing these will not provide much more improvemt to
model.  Given that we have added to the complexity of the nueral
network by adding two hidden layers with 300 neurons each, I do not
expect to find much improvement by this course of action.  This is
merely reccomended so as to be systematic in testing.

2.  Conduct PCA of original dataset to find more variables that may
predict depression better.
```

Comparing from last weeks assignment, the results of this model confirm that it was not the analytical technique is not the issue. Niether techinique produced different results. It may be that poverty, general health, and smoking habits are not enough for predicting depression. This implies that we need to add more variables, hence why one of the reccomndations is to conduct a PCA of the original data.

I must further note that there is published research that shows that poverty does predict depression. The results of this model do not contradict this. This model is using depression as well as general health, and smoking habits. It is the three variables combined that produces a model that has trouble distinguishing "not-depressed" from "depressed".

When you combine the knowledge from published resaerch on poverty and depression, and the results of this model, we can more firmly conclude that we need more variables to better predict depression.