In [1]:
```
#Libraries imported
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import Accuracy
```

In [3]:
```
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report, accuracy_score,roc_curve,confusion_matri
```

In [4]:
```
# Load the training dataset
instagram_df_train=pd.read_csv('E://insta_train.csv')
instagram_df_train
```

Out[4]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follov
0	1	0.27	0	0.00	0	53	0	0	32	1
1	1	0.00	2	0.00	0	44	0	0	286	2
2	1	0.10	2	0.00	0	0	0	1	13	
3	1	0.00	1	0.00	0	82	0	0	679	
4	1	0.00	2	0.00	0	0	0	1	6	
...	...	...	...	...	...	...	...	...	...	
571	1	0.55	1	0.44	0	0	0	0	33	
572	1	0.38	1	0.33	0	21	0	0	44	
573	1	0.57	2	0.00	0	0	0	0	4	
574	1	0.57	1	0.00	0	11	0	0	0	
575	1	0.27	1	0.00	0	0	0	0	2	

576 rows × 12 columns

In [5]:

```python
# Load the testing data
instagram_df_test=pd.read_csv('E://insta_test.csv')
instagram_df_test
```

Out[5]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follow
0	1	0.33	1	0.33	1	30	0	1	35	
1	1	0.00	5	0.00	0	64	0	1	3	
2	1	0.00	2	0.00	0	82	0	1	319	
3	1	0.00	1	0.00	0	143	0	1	273	14
4	1	0.50	1	0.00	0	76	0	1	6	
...	...	...	...	...	...	...	...	...	...	...
115	1	0.29	1	0.00	0	0	0	0	13	
116	1	0.40	1	0.00	0	0	0	0	4	
117	1	0.00	2	0.00	0	0	0	0	3	
118	0	0.17	1	0.00	0	0	0	0	1	
119	1	0.44	1	0.00	0	0	0	0	3	

120 rows × 12 columns

In [6]:

```python
instagram_df_train.head()
```

Out[6]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followe
0	1	0.27	0	0.0	0	53	0	0	32	100
1	1	0.00	2	0.0	0	44	0	0	286	274
2	1	0.10	2	0.0	0	0	0	1	13	15
3	1	0.00	1	0.0	0	82	0	0	679	41
4	1	0.00	2	0.0	0	0	0	1	6	15

In [7]:

```python
instagram_df_train.tail()
```

Out[7]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follov
571	1	0.55	1	0.44	0	0	0	0	33	
572	1	0.38	1	0.33	0	21	0	0	44	
573	1	0.57	2	0.00	0	0	0	0	4	
574	1	0.57	1	0.00	0	11	0	0	0	
575	1	0.27	1	0.00	0	0	0	0	2	

In [8]:

```
instagram_df_test.head()
```

Out[8]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#followe
0	1	0.33	1	0.33	1	30	0	1	35	48
1	1	0.00	5	0.00	0	64	0	1	3	3
2	1	0.00	2	0.00	0	82	0	1	319	32
3	1	0.00	1	0.00	0	143	0	1	273	1489
4	1	0.50	1	0.00	0	76	0	1	6	22

In [9]:

```
instagram_df_test.tail()
```

Out[9]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follow
115	1	0.29	1	0.0	0	0	0	0	13	
116	1	0.40	1	0.0	0	0	0	0	4	
117	1	0.00	2	0.0	0	0	0	0	3	
118	0	0.17	1	0.0	0	0	0	0	1	
119	1	0.44	1	0.0	0	0	0	0	3	

Performing Exploratory Data Analysis EDA

In [10]:

```
# Getting dataframe info
instagram_df_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 576 entries, 0 to 575
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   profile pic           576 non-null    int64  
 1   nums/length username  576 non-null    float64
 2   fullname words        576 non-null    int64  
 3   nums/length fullname  576 non-null    float64
 4   name==username        576 non-null    int64  
 5   description length    576 non-null    int64  
 6   external URL          576 non-null    int64  
 7   private               576 non-null    int64  
 8   #posts                576 non-null    int64  
 9   #followers            576 non-null    int64  
 10  #follows              576 non-null    int64  
 11  fake                  576 non-null    int64  
dtypes: float64(2), int64(10)
memory usage: 54.1 KB
```

In [11]:

```
# Get the statistical summary of the dataframe
instagram_df_train.describe()
```

Out[11]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private
count	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000	576.000000
mean	0.701389	0.163837	1.460069	0.036094	0.034722	22.623264	0.116319	0.381944
std	0.458047	0.214096	1.052601	0.125121	0.183234	37.702987	0.320886	0.486285
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.310000	2.000000	0.000000	0.000000	34.000000	0.000000	1.000000
max	1.000000	0.920000	12.000000	1.000000	1.000000	150.000000	1.000000	1.000000

In [12]:

```
# Checking if null values exist
instagram_df_train.isnull().sum()
```

Out[12]:

```
profile pic             0
nums/length username    0
fullname words          0
nums/length fullname    0
name==username          0
description length      0
external URL            0
private                 0
#posts                  0
#followers              0
#follows                0
fake                    0
dtype: int64
```

In [13]:

```
# Get the number of unique values in the "profile pic" feature
instagram_df_train['profile pic'].value_counts()
```

Out[13]:

```
1    404
0    172
Name: profile pic, dtype: int64
```

In [14]:

```
# Get the number of unique values in "fake" (Target column)
instagram_df_train['fake'].value_counts()
```

Out[14]:

```
0    288
1    288
Name: fake, dtype: int64
```

In [15]:

```
instagram_df_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 120 entries, 0 to 119
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   profile pic           120 non-null    int64  
 1   nums/length username  120 non-null    float64
 2   fullname words        120 non-null    int64  
 3   nums/length fullname  120 non-null    float64
 4   name==username        120 non-null    int64  
 5   description length    120 non-null    int64  
 6   external URL          120 non-null    int64  
 7   private               120 non-null    int64  
 8   #posts                120 non-null    int64  
 9   #followers            120 non-null    int64  
 10  #follows              120 non-null    int64  
 11  fake                  120 non-null    int64  
dtypes: float64(2), int64(10)
memory usage: 11.4 KB
```

In [16]:

```
instagram_df_test.describe()
```

Out[16]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private
count	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000	120.000000
mean	0.758333	0.179917	1.550000	0.071333	0.041667	27.200000	0.100000	0.308333
std	0.429888	0.241492	1.187116	0.209429	0.200664	42.588632	0.301258	0.463741
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	1.000000	0.330000	2.000000	0.000000	0.000000	45.250000	0.000000	1.000000
max	1.000000	0.890000	9.000000	1.000000	1.000000	149.000000	1.000000	1.000000

In [17]:

```
instagram_df_test.isnull().sum()
```

Out[17]:

```
profile pic             0
nums/length username    0
fullname words          0
nums/length fullname    0
name==username          0
description length      0
external URL            0
private                 0
#posts                  0
#followers              0
#follows                0
fake                    0
dtype: int64
```

In [18]:

```
instagram_df_test['fake'].value_counts()
```

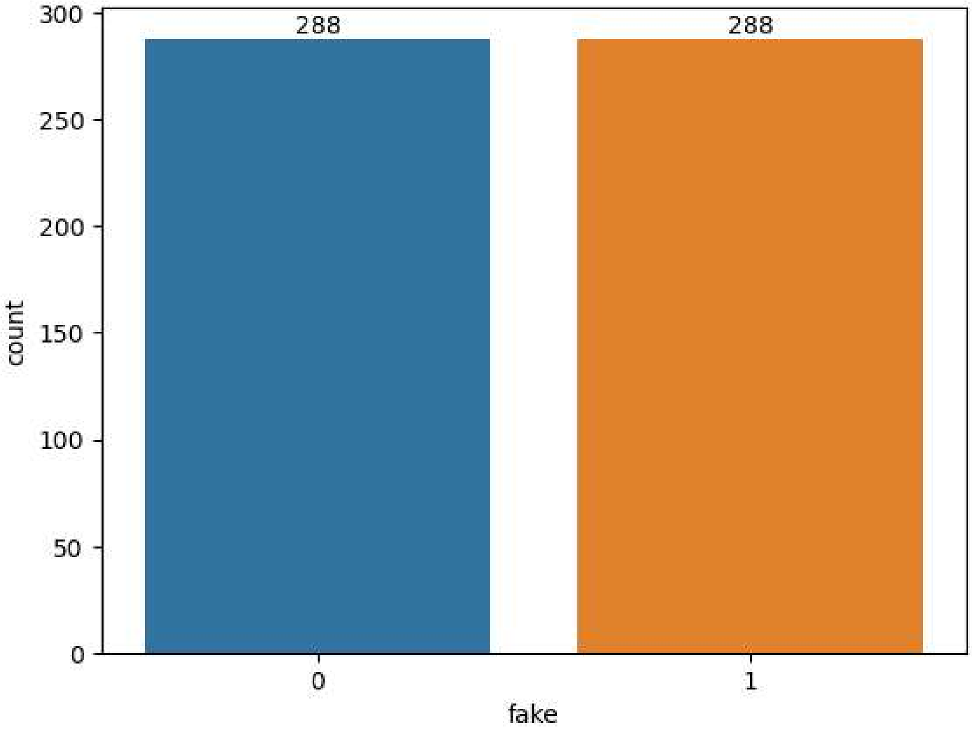
Out[18]:

```
0    60
1    60
Name: fake, dtype: int64
```

Perform Data Visualizations

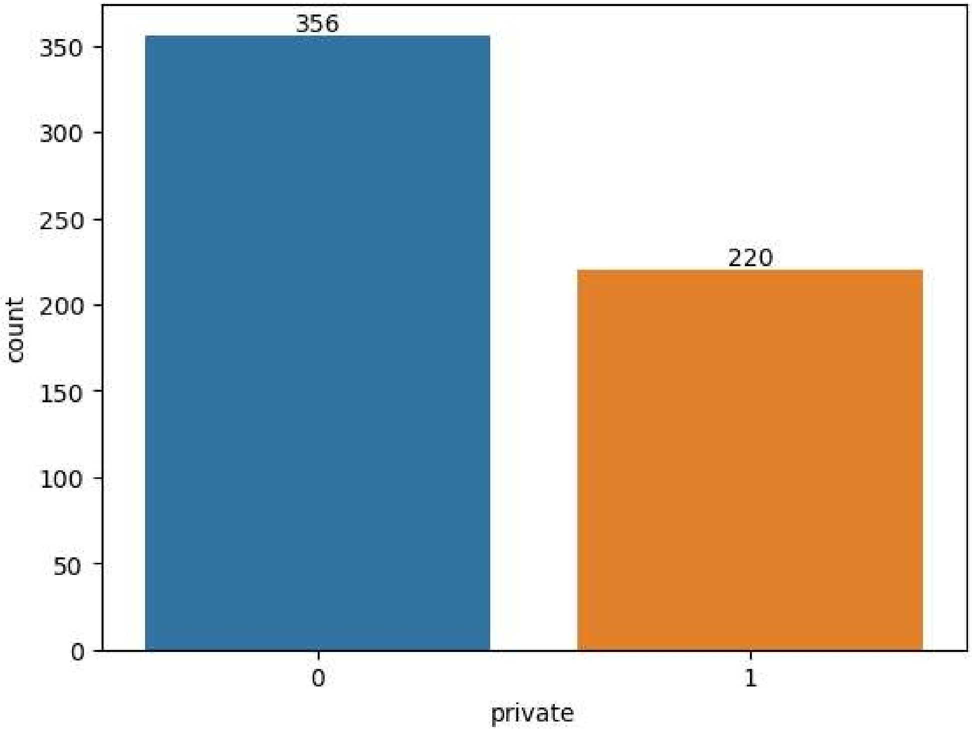
In [19]:

```
# Visualize the data
ax=sns.countplot(instagram_df_train['fake'])
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```



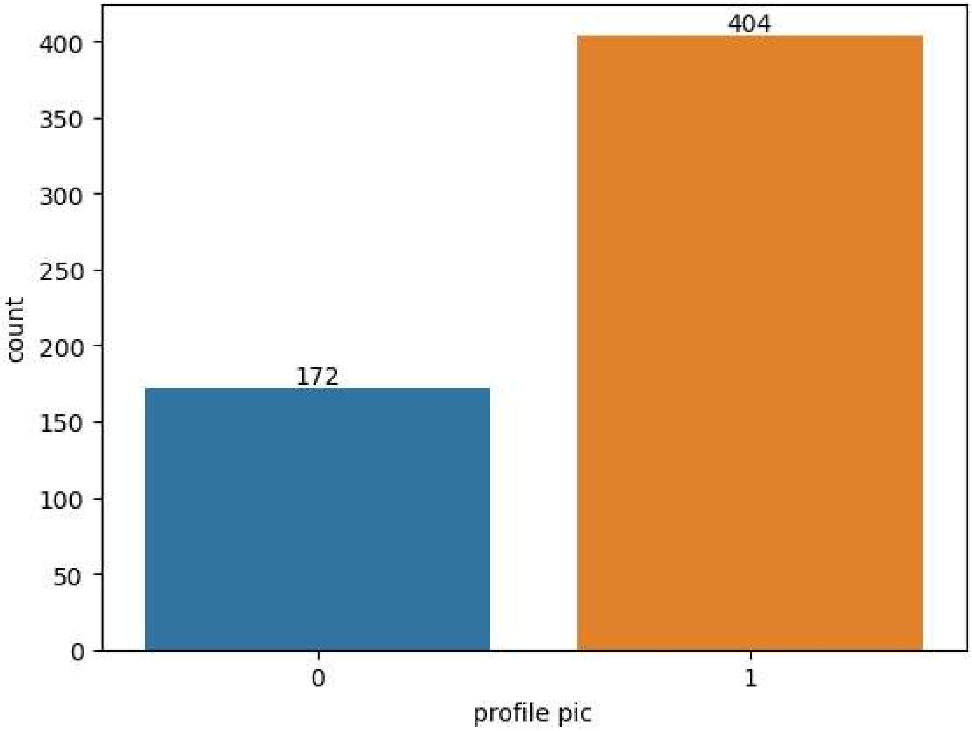
In [20]:

```
# Visualize the private column data
ax=sns.countplot(instagram_df_train['private'])
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```



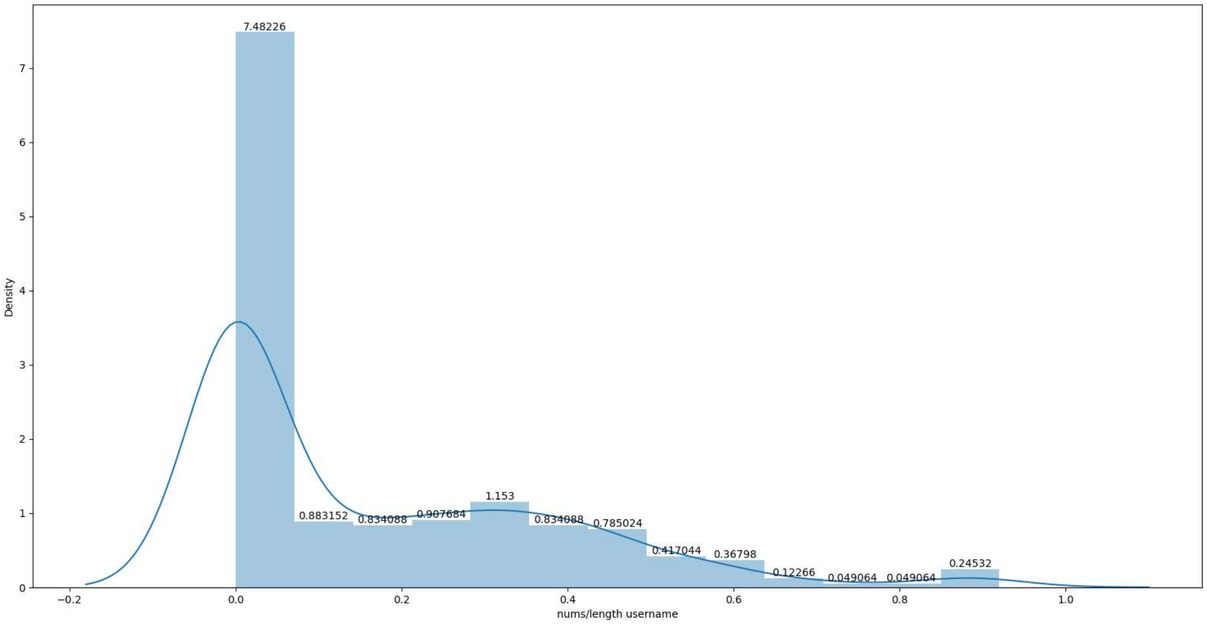
In [21]:

```
# Visualize the "profile pic" column data
ax=sns.countplot(instagram_df_train['profile pic'])
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```



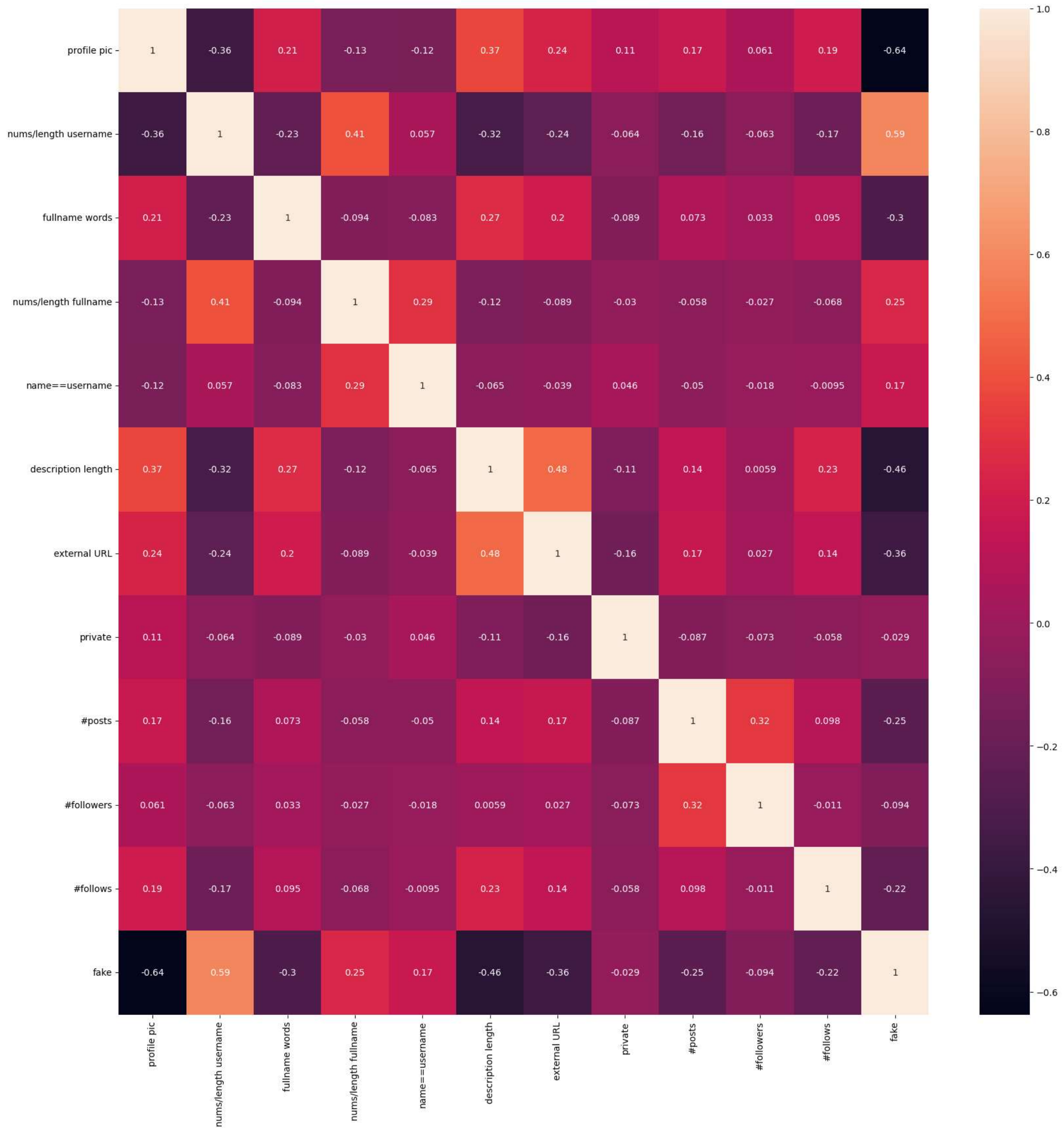
In [22]:

```
# Visualize the data
plt.figure(figsize = (20, 10))
ax=sns.distplot(instagram_df_train['nums/length username'])
for i in ax.containers:
    ax.bar_label(i)
plt.show()
```



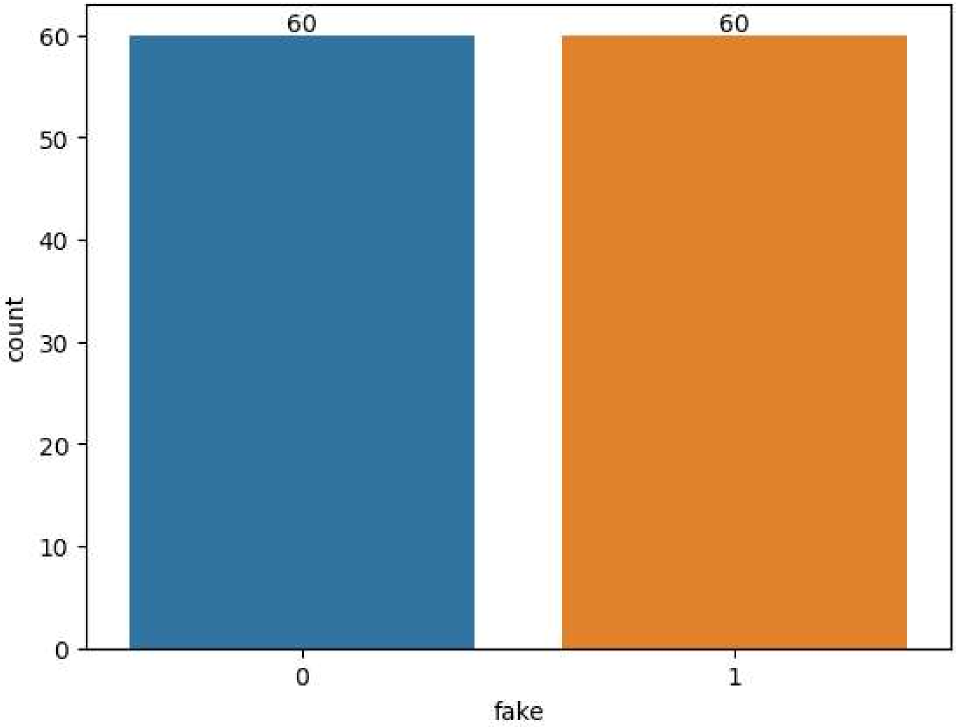
In [23]:

```
# Correlation plot
plt.figure(figsize=(20, 20))
cm = instagram_df_train.corr()
ax = plt.subplot()
# heatmap for correlation matrix
sns.heatmap(cm, annot = True, ax = ax)
plt.show()
```



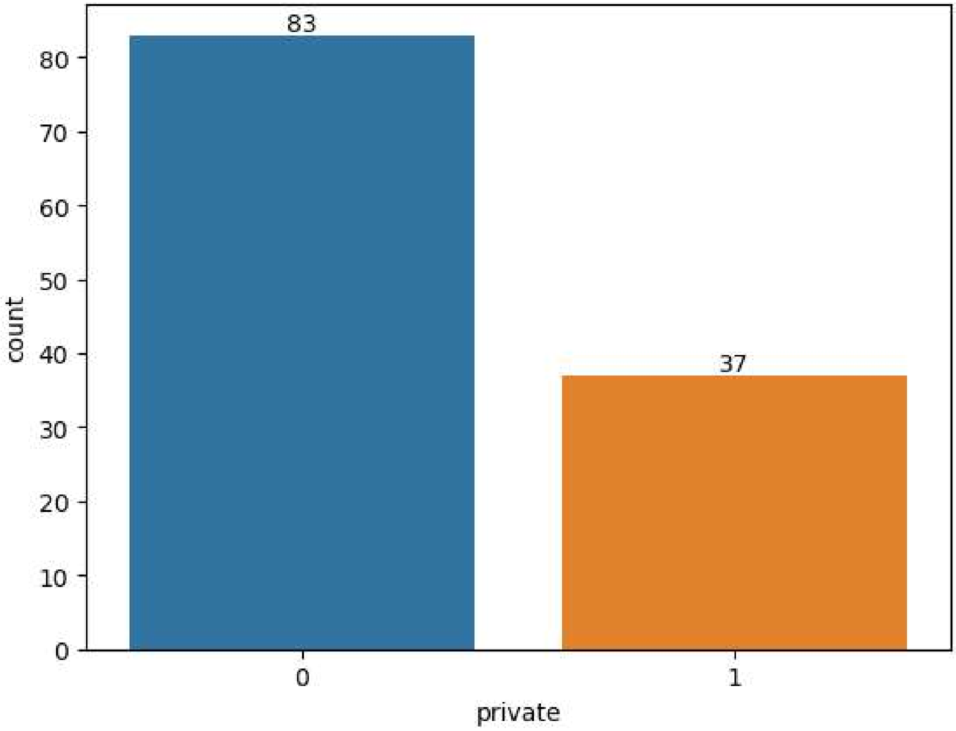
In [53]:

```
ax=sns.countplot(instagram_df_test['fake'])
for i in ax.containers:
    ax.bar_label(i)
```



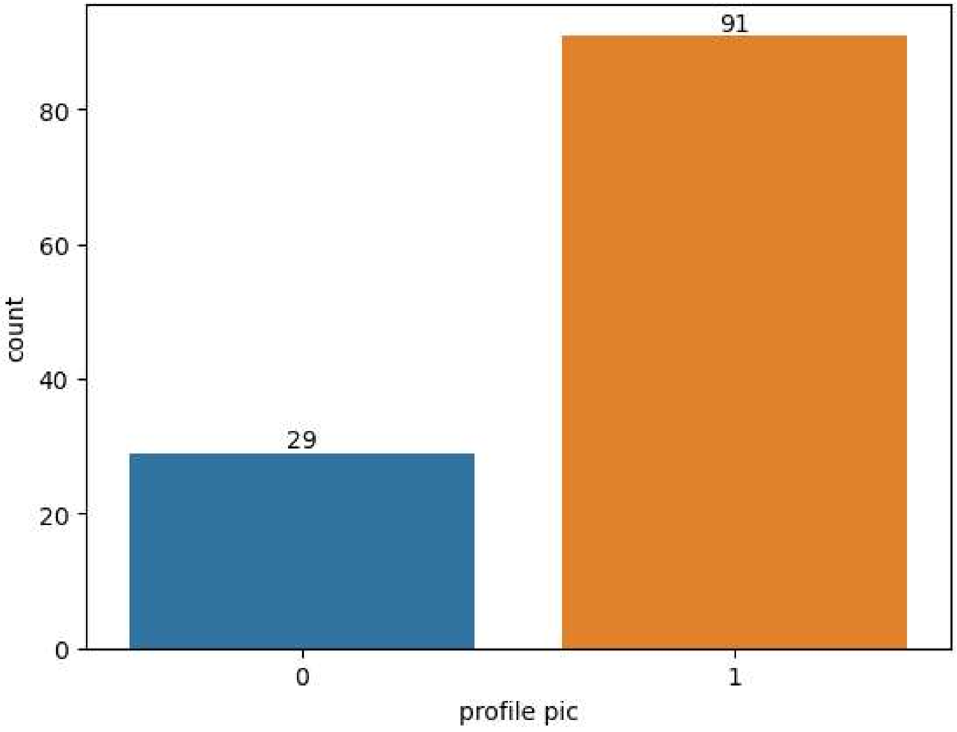
In [54]:

```
ax=sns.countplot(instagram_df_test['private'])
for i in ax.containers:
    ax.bar_label(i)
```



In [55]:

```
ax=sns.countplot(instagram_df_test['profile pic'])
for i in ax.containers:
    ax.bar_label(i)
```



Preparing Data to Train the Model

In [27]:

```
# Training and testing dataset (inputs)
X_train = instagram_df_train.drop(columns = ['fake'])
X_test = instagram_df_test.drop(columns = ['fake'])
X_train
```

Out[27]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follov
0	1	0.27	0	0.00	0	53	0	0	32	1
1	1	0.00	2	0.00	0	44	0	0	286	2
2	1	0.10	2	0.00	0	0	0	1	13	
3	1	0.00	1	0.00	0	82	0	0	679	
4	1	0.00	2	0.00	0	0	0	1	6	
...	...	...	...	...	...	...	...	...	...	
571	1	0.55	1	0.44	0	0	0	0	33	
572	1	0.38	1	0.33	0	21	0	0	44	
573	1	0.57	2	0.00	0	0	0	0	4	
574	1	0.57	1	0.00	0	11	0	0	0	
575	1	0.27	1	0.00	0	0	0	0	2	

576 rows × 11 columns

```
In [28]:  X_test
```

Out[28]:

	profile pic	nums/length username	fullname words	nums/length fullname	name==username	description length	external URL	private	#posts	#follow
0	1	0.33	1	0.33	1	30	0	1	35	
1	1	0.00	5	0.00	0	64	0	1	3	
2	1	0.00	2	0.00	0	82	0	1	319	
3	1	0.00	1	0.00	0	143	0	1	273	14
4	1	0.50	1	0.00	0	76	0	1	6	
...	...	...	...	...	...	...	...	...	...	
115	1	0.29	1	0.00	0	0	0	0	13	
116	1	0.40	1	0.00	0	0	0	0	4	
117	1	0.00	2	0.00	0	0	0	0	3	
118	0	0.17	1	0.00	0	0	0	0	1	
119	1	0.44	1	0.00	0	0	0	0	3	

120 rows × 11 columns

```
In [29]:  # Training and testing dataset (Outputs)
          y_train = instagram_df_train['fake']
          y_test = instagram_df_test['fake']
```

```
In [30]:  y_train
```

```
Out[30]: 0      0
         1      0
         2      0
         3      0
         4      0
               ..
         571    1
         572    1
         573    1
         574    1
         575    1
         Name: fake, Length: 576, dtype: int64
```

```
In [31]:  y_test
```

```
Out[31]: 0      0
         1      0
         2      0
         3      0
         4      0
               ..
         115    1
         116    1
         117    1
         118    1
         119    1
         Name: fake, Length: 120, dtype: int64
```

Scale the data before training the model

In [32]:

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler
scaler_x=StandardScaler()
X_train=scaler_x.fit_transform(X_train)
X_test=scaler_x.transform(X_test)
```

In [33]:

```
y_train = tf.keras.utils.to_categorical(y_train, num_classes = 2)
y_test = tf.keras.utils.to_categorical(y_test, num_classes = 2)
```

In [34]:

```
y_train
```

Out[34]:

```
array([[1., 0.],
       [1., 0.],
       [1., 0.],
       ...,
       [0., 1.],
       [0., 1.],
       [0., 1.]], dtype=float32)
```

In [35]:

```
y_test
```

```
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
       [1., 0.],
```

In [36]:

```
# print the shapes of training and testing datasets
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[36]:

```
((576, 11), (120, 11), (576, 2), (120, 2))
```

In [37]:

```
Training_data = len(X_train)/( len(X_test) + len(X_train) ) * 100
Training_data
```

Out[37]:

```
82.75862068965517
```

In [38]:

```
Testing_data = len(X_test)/( len(X_test) + len(X_train) ) * 100
Testing_data
```

Out[38]:

```
17.24137931034483
```

Building and Training Deep Training Model

In [39]:

```
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [40]:

```
model = Sequential()
model.add(Dense(50, input_dim=11, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(2,activation='softmax'))
```

In [41]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 50)                600       
                                                                 
 dense_1 (Dense)             (None, 150)               7650      
                                                                 
 dropout (Dropout)           (None, 150)               0         
                                                                 
 dense_2 (Dense)             (None, 150)               22650     
                                                                 
 dropout_1 (Dropout)         (None, 150)               0         
                                                                 
 dense_3 (Dense)             (None, 25)                3775      
                                                                 
 dropout_2 (Dropout)         (None, 25)                0         
                                                                 
 dense_4 (Dense)             (None, 2)                 52        
                                                                 
=================================================================
Total params: 34,727
Trainable params: 34,727
Non-trainable params: 0
_________________________________________________________________
```

In [42]:

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

epochs_hist = model.fit(X_train, y_train, epochs = 50,  verbose = 1, validation_split = 0.1
```

```
- val_loss: 0.1856 - val_accuracy: 0.9138
Epoch 7/50
17/17 [==============================] - 0s 11ms/step - loss: 0.2355 - accuracy: 0.9151
- val_loss: 0.1857 - val_accuracy: 0.8966
Epoch 8/50
17/17 [==============================] - 0s 11ms/step - loss: 0.2024 - accuracy: 0.9189
- val_loss: 0.1674 - val_accuracy: 0.9138
Epoch 9/50
17/17 [==============================] - 0s 11ms/step - loss: 0.2145 - accuracy: 0.9324
- val_loss: 0.2151 - val_accuracy: 0.8966
Epoch 10/50
17/17 [==============================] - 0s 13ms/step - loss: 0.2018 - accuracy: 0.9247
- val_loss: 0.1952 - val_accuracy: 0.8966
Epoch 11/50
17/17 [==============================] - 0s 14ms/step - loss: 0.1915 - accuracy: 0.9228
- val_loss: 0.2250 - val_accuracy: 0.8966
Epoch 12/50
17/17 [==============================] - 0s 13ms/step - loss: 0.1933 - accuracy: 0.9286
- val_loss: 0.1692 - val_accuracy: 0.9310
Epoch 13/50
```

In [43]:

```
import tensorflow.keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

In [44]:

```
model = Sequential()
model.add(Dense(50, input_dim=11, activation='relu'))
model.add(Dense(150, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(25, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(2, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 50)                600

 dense_6 (Dense)             (None, 150)               7650

 dropout_3 (Dropout)         (None, 150)               0

 dense_7 (Dense)             (None, 25)                3775

 dropout_4 (Dropout)         (None, 25)                0

 dense_8 (Dense)             (None, 25)                650

 dropout_5 (Dropout)         (None, 25)                0

 dense_9 (Dense)             (None, 2)                 52

=================================================================
Total params: 12,727
Trainable params: 12,727
Non-trainable params: 0
_________________________________________________________________
```

Access the Performance of the model

In [45]:

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

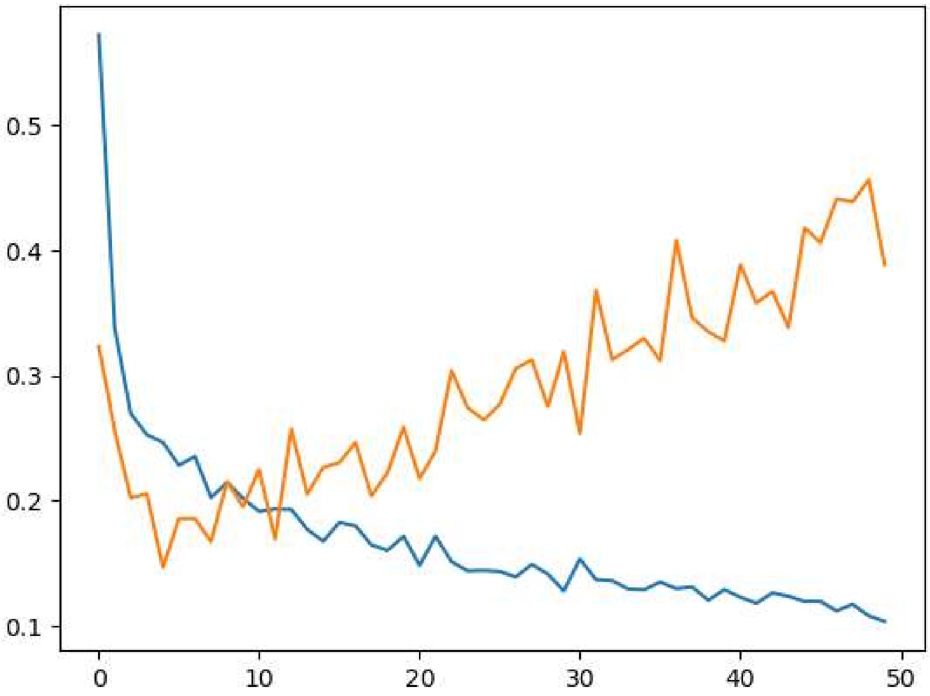
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [46]:

```
plt.plot(epochs_hist.history['loss'])
plt.plot(epochs_hist.history['val_loss'])
```

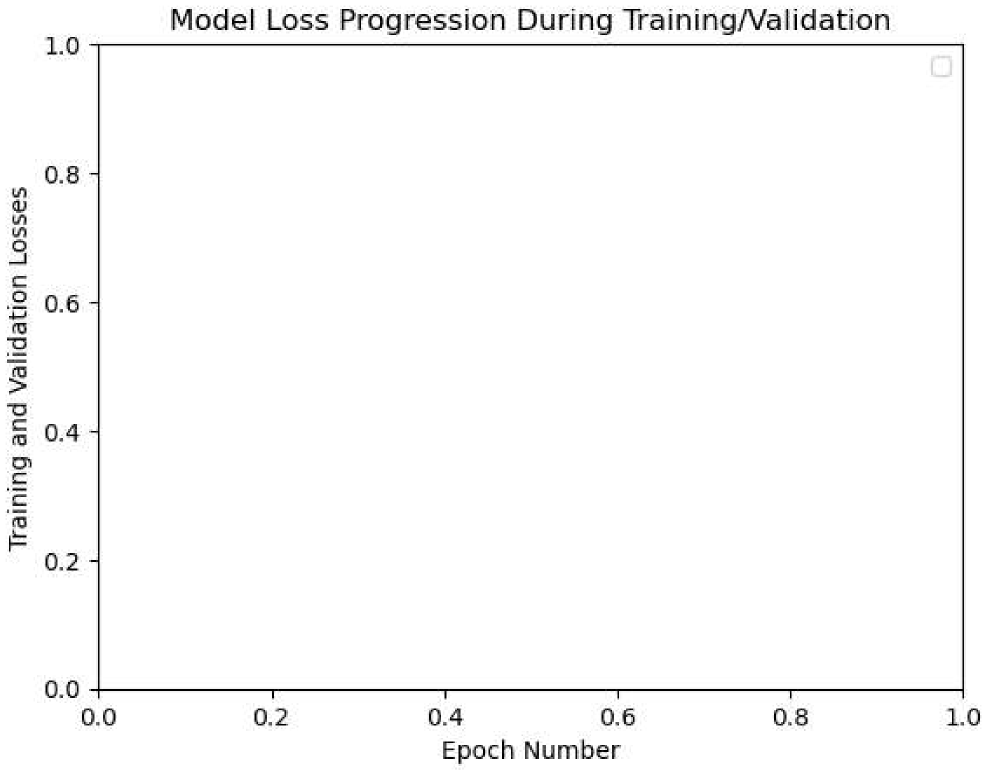
Out[46]:

```
[<matplotlib.lines.Line2D at 0x24edf290610>]
```



In [47]:

```
plt.title('Model Loss Progression During Training/Validation')
plt.ylabel('Training and Validation Losses')
plt.xlabel('Epoch Number')
plt.legend(['Training Loss', 'Validation Loss'])
plt.show()
```



In [48]:

```
predicted = model.predict(X_test)
```

```
4/4 [==============================] - 0s 4ms/step
```

In [49]:

```
predicted_value = []
test = []
for i in predicted:
    predicted_value.append(np.argmax(i))
```

In [50]:

```
for i in y_test:
    test.append(np.argmax(i))
```

In [51]:

```
print(classification_report(test, predicted_value))
```

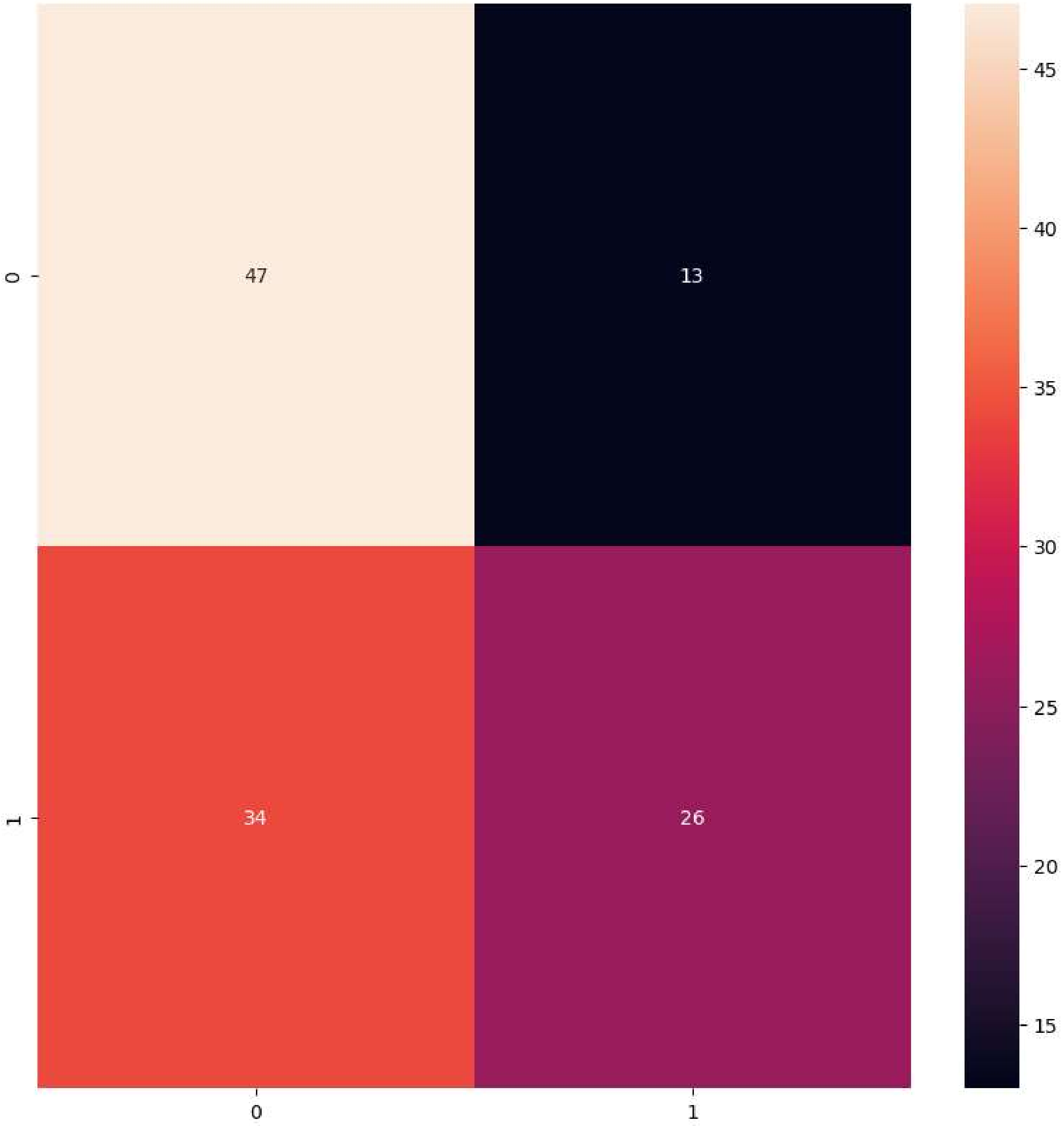
```
              precision    recall  f1-score   support

           0       0.58      0.78      0.67        60
           1       0.67      0.43      0.53        60

    accuracy                           0.61       120
   macro avg       0.62      0.61      0.60       120
weighted avg       0.62      0.61      0.60       120
```

In [52]:

```
plt.figure(figsize=(10, 10))
cm=confusion_matrix(test, predicted_value)
sns.heatmap(cm, annot=True)
plt.show()
```



In [ ]: