In [6]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [8]:
```
df=pd.read_csv("data.csv")
df
```

Out[8]:

	process.b1.capacity	process.b2.capacity	process.b3.capacity	process.b4
0	0	0	2	
1	0	0	2	
2	0	0	2	
3	0	0	2	
4	0	0	2	
...	...	...	...	
2038	2	3	2	
2039	2	3	2	
2040	2	3	2	
2041	2	3	2	
2042	2	3	2	

2043 rows × 9 columns

In [9]:
```
# Top 5 values in the data

df.head()
```

Out[9]:

	process.b1.capacity	process.b2.capacity	process.b3.capacity	process.b4.cap
0	0	0	2	
1	0	0	2	
2	0	0	2	
3	0	0	2	
4	0	0	2	

In [10]:
```
# Bottom 5 values of the data

df.tail()
```

Out[10]:

	process.b1.capacity	process.b2.capacity	process.b3.capacity	process.b4
2038	2	3	2	
2039	2	3	2	
2040	2	3	2	
2041	2	3	2	
2042	2	3	2	

In [12]:
```
df.shape
```
Out[12]:
```
(2043, 9)
```

In [13]:
```
df.columns
```
Out[13]:
```
Index(['process.b1.capacity', 'process.b2.capacity', 'process.b3.capacity',
       'process.b4.capacity', 'property.price', 'property.product',
       'property.winner', 'verification.result', 'verification.time'],
      dtype='object')
```

In [14]:
```
# To get the information of the data

df.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2043 entries, 0 to 2042
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   process.b1.capacity  2043 non-null   int64
 1   process.b2.capacity  2043 non-null   int64
 2   process.b3.capacity  2043 non-null   int64
 3   process.b4.capacity  2043 non-null   int64
 4   property.price       2043 non-null   int64
 5   property.product     2043 non-null   int64
 6   property.winner      2043 non-null   int64
 7   verification.result  2043 non-null   bool
 8   verification.time    2043 non-null   float64
dtypes: bool(1), float64(1), int64(7)
memory usage: 129.8 KB
```

In [15]:
```
# To find the sum of the null values

df.isnull().sum()
```

Out[15]:

	0
process.b1.capacity	0
process.b2.capacity	0
process.b3.capacity	0
process.b4.capacity	0
property.price	0
property.product	0
property.winner	0
verification.result	0
verification.time	0

dtype: int64

In [17]:
```
df.describe()
```

Out[17]:

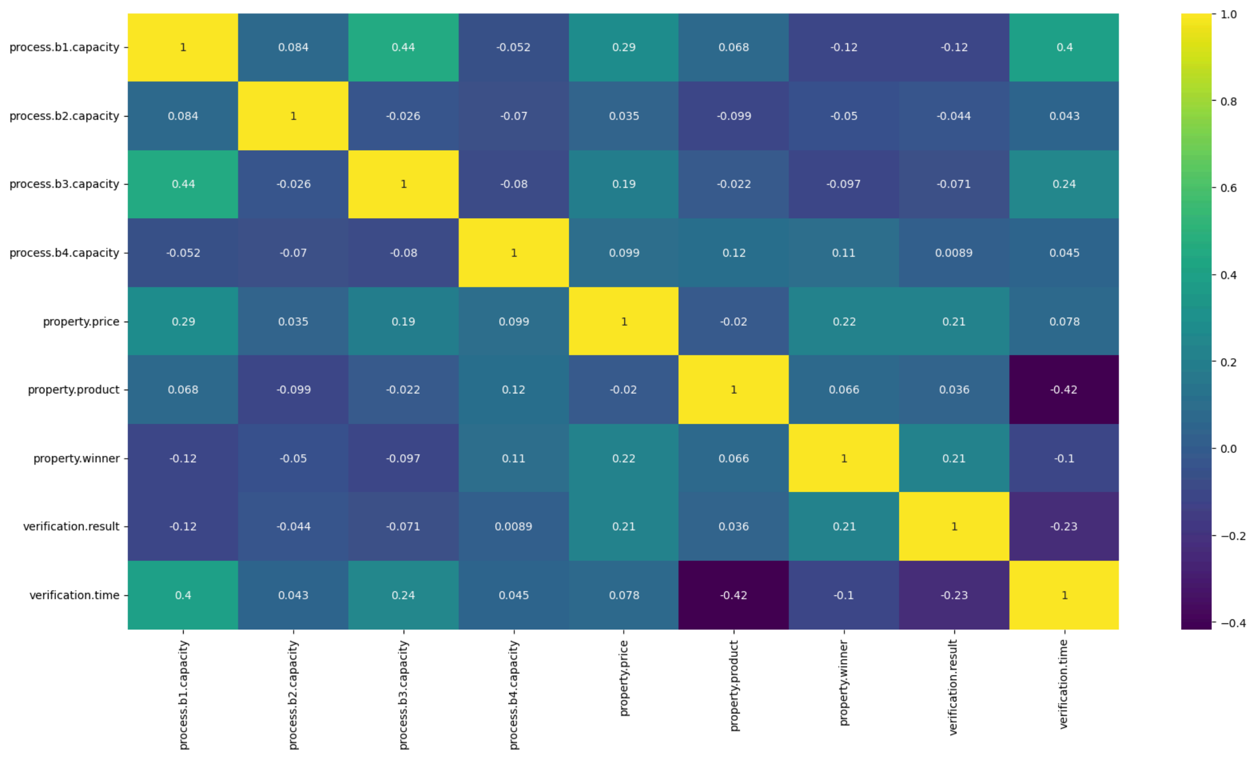
	process.b1.capacity	process.b2.capacity	process.b3.capacity	process.b
count	2043.000000	2043.000000	2043.000000	20
mean	1.000000	2.093979	1.883994	
std	0.816696	0.811269	0.320310	
min	0.000000	0.000000	1.000000	
25%	0.000000	1.000000	2.000000	
50%	1.000000	2.000000	2.000000	
75%	2.000000	3.000000	2.000000	
max	2.000000	3.000000	2.000000	

In [18]:
```
df.corr()
```

Out[18]:

	process.b1.capacity	process.b2.capacity	process.b3.capaci
process.b1.capacity	1.000000	0.084260	0.4436
process.b2.capacity	0.084260	1.000000	-0.0258
process.b3.capacity	0.443671	-0.025869	1.0000
process.b4.capacity	-0.052370	-0.069726	-0.0797
property.price	0.285558	0.035033	0.1894
property.product	0.068131	-0.099167	-0.0222
property.winner	-0.121864	-0.049640	-0.0965
verification.result	-0.120126	-0.044442	-0.0713
verification.time	0.398359	0.042732	0.2400

In [21]:
```python
# heatmap of the features
plt.figure(figsize=(20,10))
sns.heatmap(df.corr(),annot=True,cmap="viridis")
plt.show()
```



In [24]:
```python
x=df.drop("verification.result",axis=1)
y=df["verification.result"]
```

In [26]:
```python
from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(x,y)
```

Out[26]:

```
▾ ExtraTreesRegressor
ExtraTreesRegressor()
```

In [27]:

```
x.head()
```

Out[27]:

	process.b1.capacity	process.b2.capacity	process.b3.capacity	process.b4.cap
0	0	0	2	
1	0	0	2	
2	0	0	2	
3	0	0	2	
4	0	0	2	

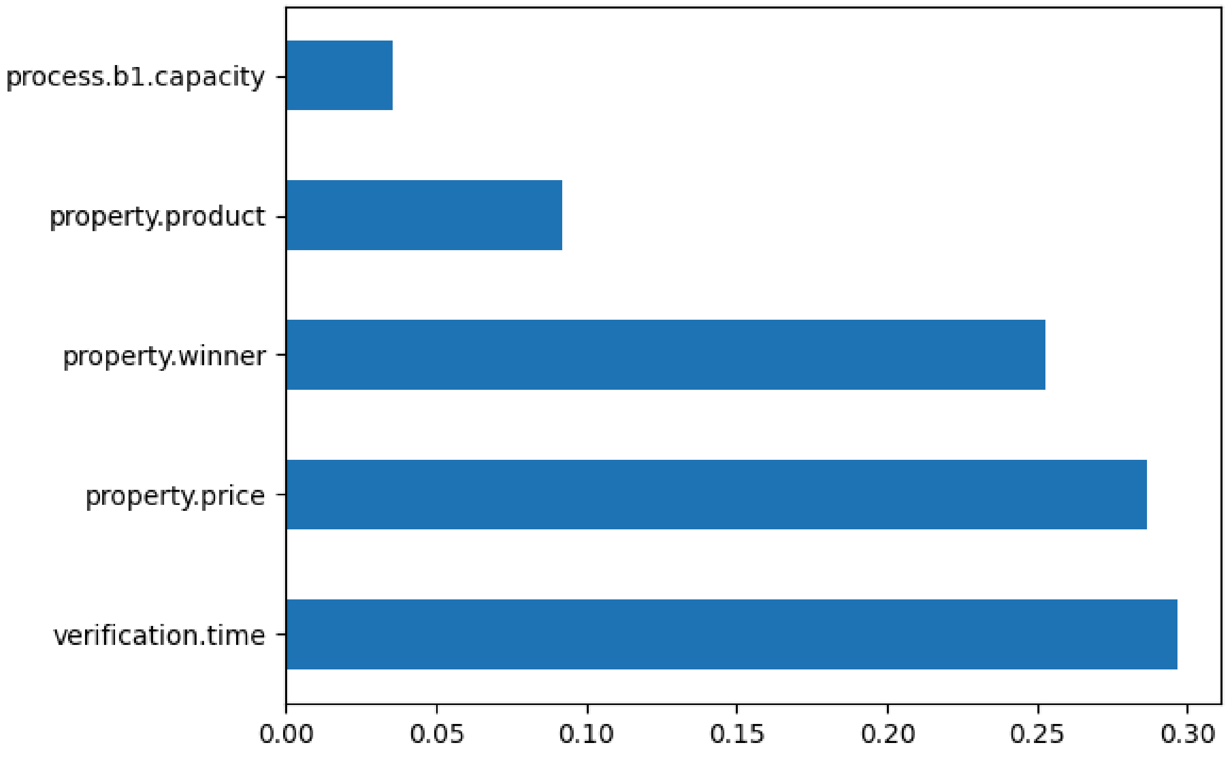
In [29]:

```
print(model.feature_importances_)
```

```
[0.03567303 0.01846186 0.00233455 0.01521585 0.28661194 0.09216356
 0.25294381 0.2965954 ]
```

In [30]:

```
feat_importances=pd.Series(model.feature_importances_, index=x.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [31]:

```
# 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=
```

In [33]:
```
# Feature_scaling

from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

In [34]:
```
# importing tensorflow

import tensorflow as tf
```

In [36]:
```
ann=tf.keras.models.Sequential()
```

In [39]:
```
ann.add(tf.keras.layers.Dense(32, activation="relu"))

ann.add(tf.keras.layers.Dense(32, activation="relu"))

ann.add(tf.keras.layers.Dense(1, activation="sigmoid"))
```

In [44]:
```
#Compile the network

ann.compile(optimizer="adam", loss="binary_crossentropy", metrics=["accuracy"]
```

In [45]:
```
ann.fit(x_train, y_train, batch_size=32, epochs=100)
```

```
Epoch 1/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 4s 3ms/step - accuracy: 0.5681 - loss: 0.6464
Epoch 2/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8659 - loss: 0.3715
Epoch 3/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8737 - loss: 0.3172
Epoch 4/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.8765 - loss: 0.2965
Epoch 5/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8794 - loss: 0.2771
Epoch 6/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8736 - loss: 0.2789
Epoch 7/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8891 - loss: 0.2540
Epoch 8/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8928 - loss: 0.2566
Epoch 9/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9015 - loss: 0.2361
Epoch 10/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8897 - loss: 0.2481
Epoch 11/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8908 - loss: 0.2450
Epoch 12/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9096 - loss: 0.2235
Epoch 13/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9040 - loss: 0.2319
Epoch 14/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9083 - loss: 0.2247
Epoch 15/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9099 - loss: 0.2250
Epoch 16/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9072 - loss: 0.2206
Epoch 17/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9138 - loss: 0.2010
Epoch 18/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9083 - loss: 0.2125
Epoch 19/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9182 - loss: 0.2030
Epoch 20/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9159 - loss: 0.2098
Epoch 21/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9216 - loss: 0.1913
Epoch 22/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9254 - loss: 0.2004
Epoch 23/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9213 - loss: 0.1895
Epoch 24/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9298 - loss: 0.1865
Epoch 25/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9235 - loss: 0.2021
Epoch 26/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9271 - loss: 0.1986
Epoch 27/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9264 - loss: 0.1920
```

```
Epoch 28/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9306 - loss: 0.1840
Epoch 29/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9222 - loss: 0.1908
Epoch 30/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9232 - loss: 0.1804
Epoch 31/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9244 - loss: 0.1982
Epoch 32/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9339 - loss: 0.1762
Epoch 33/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9390 - loss: 0.1518
Epoch 34/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9345 - loss: 0.1676
Epoch 35/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9419 - loss: 0.1563
Epoch 36/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9328 - loss: 0.1759
Epoch 37/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9455 - loss: 0.1480
Epoch 38/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9348 - loss: 0.1684
Epoch 39/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9314 - loss: 0.1686
Epoch 40/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9432 - loss: 0.1545
Epoch 41/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9402 - loss: 0.1549
Epoch 42/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9471 - loss: 0.1504
Epoch 43/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9385 - loss: 0.1581
Epoch 44/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9441 - loss: 0.1479
Epoch 45/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9422 - loss: 0.1432
Epoch 46/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9278 - loss: 0.1615
Epoch 47/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9466 - loss: 0.1465
Epoch 48/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9443 - loss: 0.1559
Epoch 49/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9430 - loss: 0.1409
Epoch 50/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9463 - loss: 0.1443
Epoch 51/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9466 - loss: 0.1395
Epoch 52/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9416 - loss: 0.1472
Epoch 53/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9420 - loss: 0.1384
Epoch 54/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9407 - loss: 0.1471
```

```
Epoch 55/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9505 - loss: 0.1341
Epoch 56/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9486 - loss: 0.1431
Epoch 57/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9488 - loss: 0.1254
Epoch 58/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9393 - loss: 0.1660
Epoch 59/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9522 - loss: 0.1348
Epoch 60/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9477 - loss: 0.1298
Epoch 61/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9500 - loss: 0.1263
Epoch 62/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9552 - loss: 0.1264
Epoch 63/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9569 - loss: 0.1238
Epoch 64/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9545 - loss: 0.1238
Epoch 65/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9398 - loss: 0.1397
Epoch 66/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9473 - loss: 0.1315
Epoch 67/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9468 - loss: 0.1439
Epoch 68/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9492 - loss: 0.1309
Epoch 69/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9508 - loss: 0.1257
Epoch 70/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9554 - loss: 0.1313
Epoch 71/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9625 - loss: 0.1078
Epoch 72/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9528 - loss: 0.1210
Epoch 73/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9627 - loss: 0.1040
Epoch 74/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9558 - loss: 0.1210
Epoch 75/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9501 - loss: 0.1220
Epoch 76/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9460 - loss: 0.1271
Epoch 77/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9644 - loss: 0.1098
Epoch 78/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9685 - loss: 0.0953
Epoch 79/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9605 - loss: 0.1064
Epoch 80/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9746 - loss: 0.0906
Epoch 81/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9630 - loss: 0.1102
```

```
Epoch 82/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9554 - loss: 0.1101
Epoch 83/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9566 - loss: 0.1176
Epoch 84/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9585 - loss: 0.1129
Epoch 85/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9678 - loss: 0.0925
Epoch 86/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9670 - loss: 0.1002
Epoch 87/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9615 - loss: 0.1014
Epoch 88/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9671 - loss: 0.0951
Epoch 89/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9615 - loss: 0.1001
Epoch 90/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9558 - loss: 0.1074
Epoch 91/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9632 - loss: 0.1041
Epoch 92/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9654 - loss: 0.1004
Epoch 93/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9672 - loss: 0.0958
Epoch 94/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9715 - loss: 0.0922
Epoch 95/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9636 - loss: 0.0982
Epoch 96/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9486 - loss: 0.1219
Epoch 97/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9660 - loss: 0.0934
Epoch 98/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9669 - loss: 0.0885
Epoch 99/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step - accuracy: 0.9637 - loss: 0.0988
Epoch 100/100
52/52 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.9677 - loss: 0.0987
```

Out[45]:
```
<keras.src.callbacks.history.History at 0x7c6407b9f140>
```

In [46]:
```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_
```

Evaluation

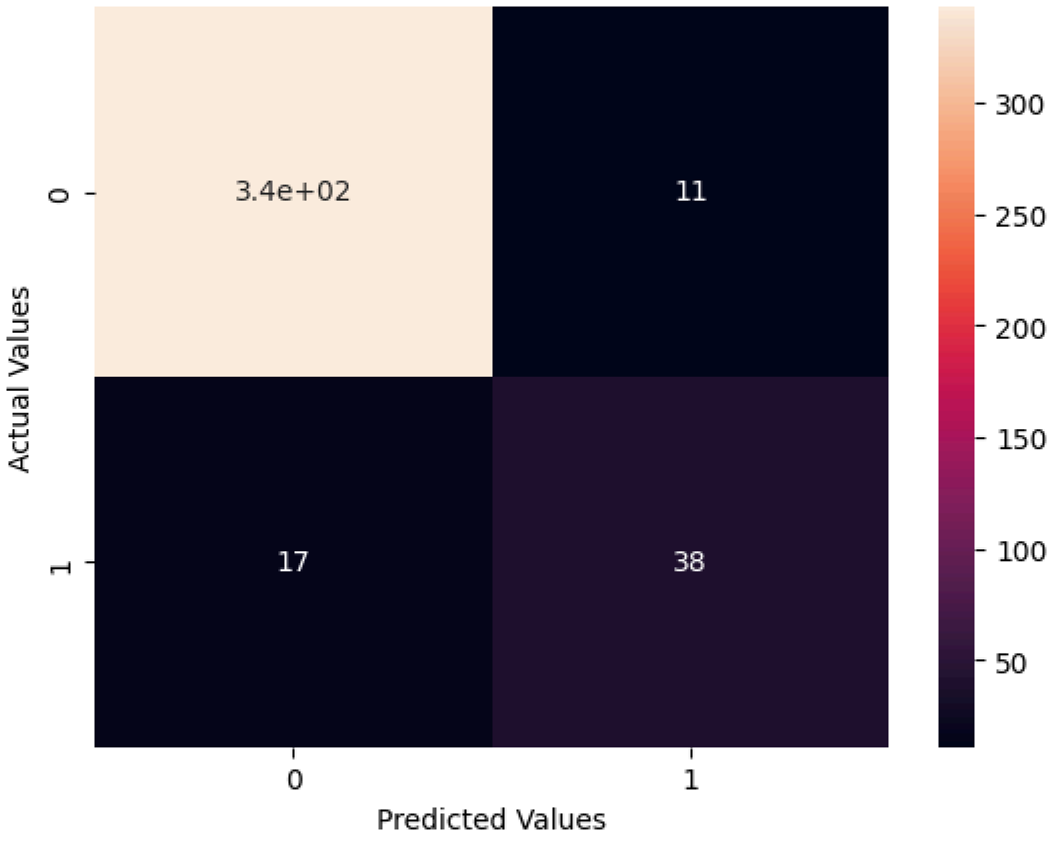
In [60]:
```python
y_pred = ann.predict(x_test)
y_pred = (y_pred > 0.5)
```

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
```

In [61]:
```python
# confusion matrix

cm=confusion_matrix(y_test,y_pred)
```

```
sns.heatmap(cm, annot=True)
plt.xlabel("Predicted Values")
plt.ylabel("Actual Values")
plt.show()
```



```
In [55]: accuracy_score(y_test,y_pred)

Out[55]: 0.9315403422982885

In [56]: classification_report(y_test,y_pred)

Out[56]: '              precision    recall  f1-score   support\n\n       False
         0.95      0.97      0.96       354\n        True       0.78      0.69
         0.73        55\n\n    accuracy                           0.93       409\n   m
         acro avg       0.86      0.83      0.85       409\nweighted avg       0.93
         0.93      0.93       409\n'
```