Collge Data

The college data set contains statistics for a large number of US colleges. There are 777 observations and 18 variables. The data is used to compare colleges both private and public based on a variety of attributes to help a similarly varied student population. There are 17 quantitative variables and 1 qualitative categorical variable labeled 'Private'. The categorical variable has 'Yes' and 'No' labels. This is a typical binary classification problem. The goal is to predict whether a college is private or public based on a combination of data attributes provided for each college.

Data source: the data set was taken from Github in csv format and is maintained on the following repository:[https://github.com/selva86/datasets/blob/master/College.csv (https://github.com/selva86/datasets/blob/master/College.csv)](https://github.com/selva86/datasets/blob/master/College.csv)

In [1]:
```
# Import all the relevant libraries
import pandas as pd
import tensorflow as tf
from tensorflow import keras
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import confusion_matrix
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
sns.set_style("whitegrid")
plt.style.use('fivethirtyeight')
```

```
Using TensorFlow backend.
```

In [3]:
```
# Load the data
df=pd.read_csv('college.csv', header=0)
```

In [4]:
```
# Dropping columns
features=list(df.columns.values)
features.remove("Private")

print(features)
```

```
['School', 'Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F_Undergrad',
'P_Undergrad', 'Outstate', 'Room_Board', 'Books', 'Personal', 'PhD', 'Terminal',
'S_F_Ratio', 'perc_alumni', 'Expend', 'Grad_Rate']
```

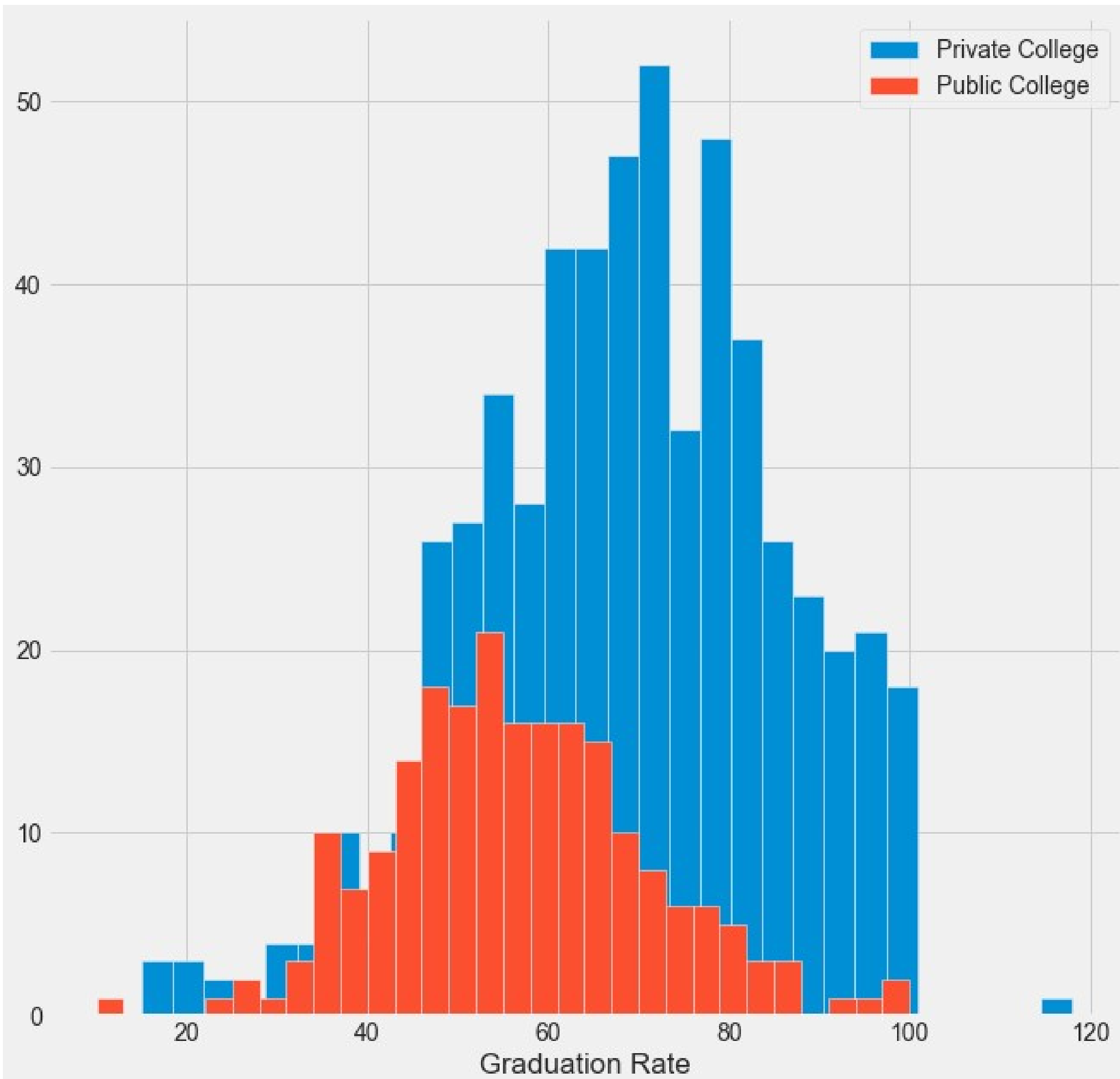
In [5]:
```
features.remove("School")
print(features)
```

```
['Apps', 'Accept', 'Enroll', 'Top10perc', 'Top25perc', 'F_Undergrad', 'P_Undergr
ad', 'Outstate', 'Room_Board', 'Books', 'Personal', 'PhD', 'Terminal', 'S_F_Rati
o', 'perc_alumni', 'Expend', 'Grad_Rate']
```

In [6]:
```
# Creating input features and target variables
X=df[features]
y=df['Private']
```

In [7]:
```python
# Correcting skewed data - Grad rate has an outlier that showed 120% for one of the schools
plt.figure(figsize=(10, 10))

df.loc[df.Private == 'Yes', 'Grad_Rate'].hist(label="Private College", bins=30)
df.loc[df.Private == 'No', 'Grad_Rate'].hist(label="Public College", bins=30)

plt.xlabel('Graduation Rate')
plt.legend()
```

Out[7]: <matplotlib.legend.Legend at 0x2093c3a58c8>



In [8]:
```python
df.loc[df.Grad_Rate > 100]
```

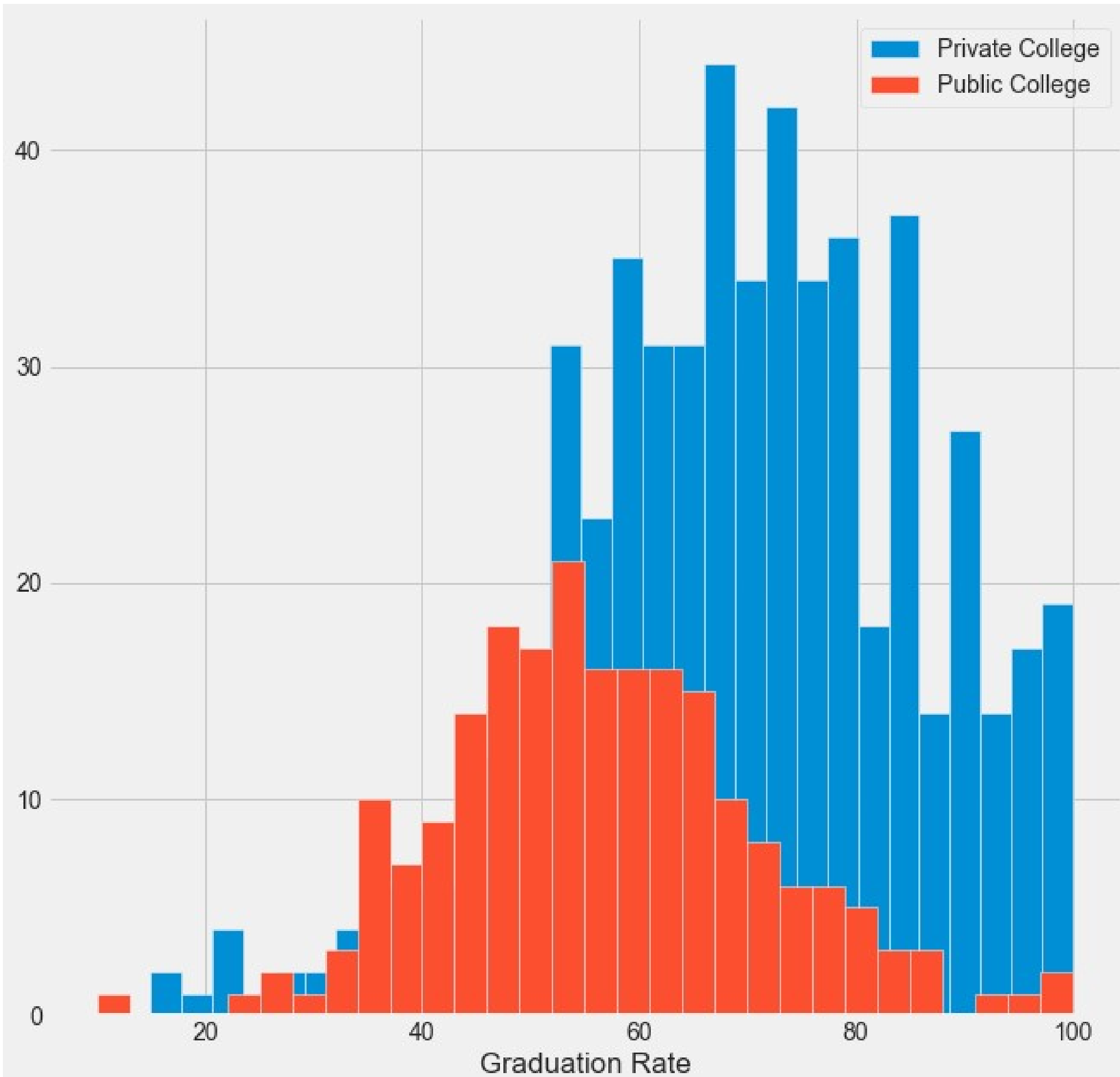
Out[8]:

	School	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F_Undergrad	P_Undergrad	Outstate	Ro
95	Cazenovia College	Yes	3847	3433	527	9	35	1010	12	9384	

In [9]:
```python
df.loc[df.Grad_Rate>100, 'Grad_Rate']=100
```

```
In [10]: plt.figure(figsize=(10, 10))

         df.loc[df.Private == 'Yes', 'Grad_Rate'].hist(label="Private College", bins=30)
         df.loc[df.Private == 'No', 'Grad_Rate'].hist(label="Public College", bins=30)

         plt.xlabel('Graduation Rate')
         plt.legend()
```

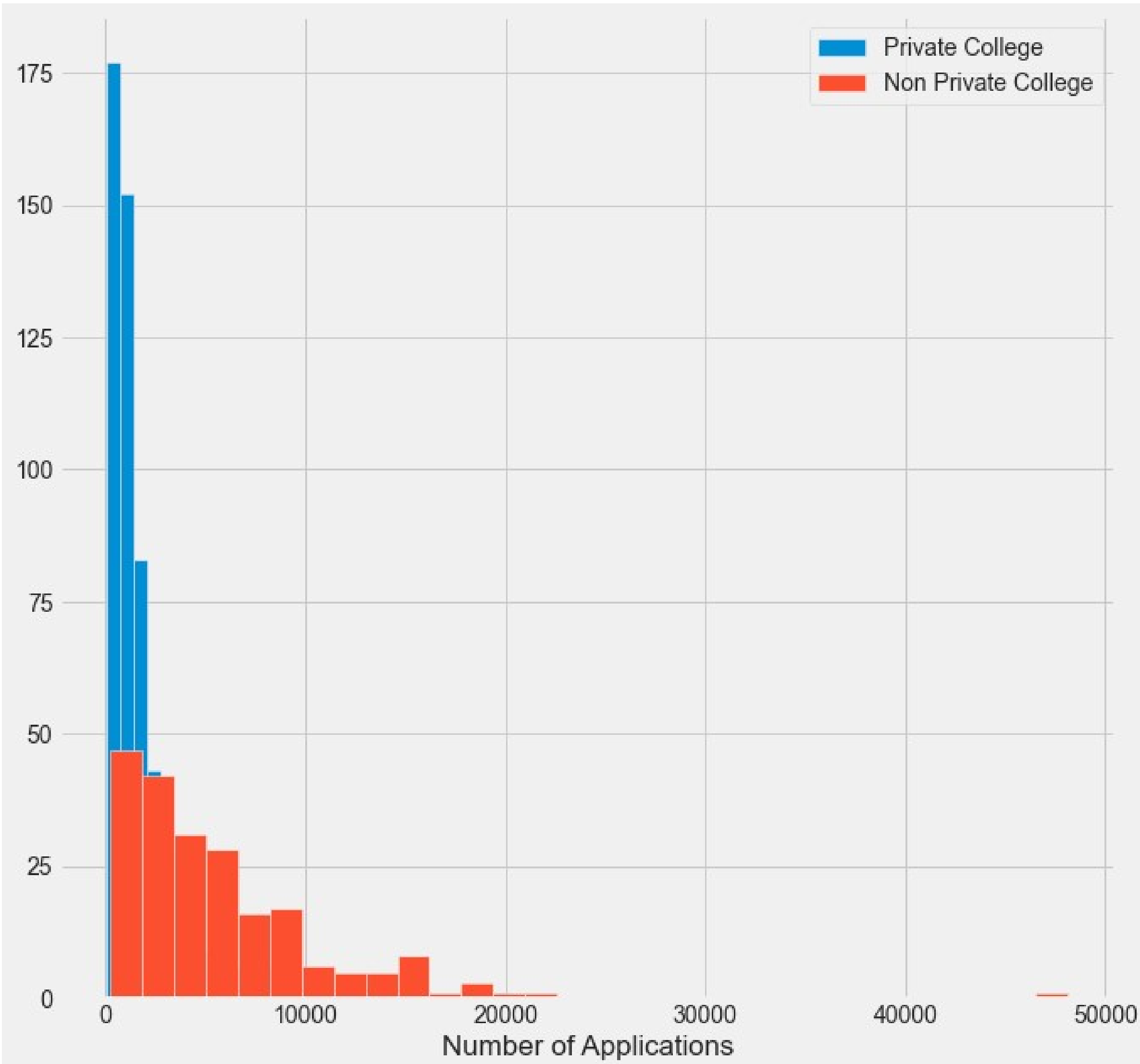
```
Out[10]: <matplotlib.legend.Legend at 0x2093c7cdb08>
```



In [11]:

```
# Checking for outliers in 'Apps'
plt.figure(figsize=(10, 10))

df.loc[df.Private == 'Yes', 'Apps'].hist(label="Private College", bins=30)
df.loc[df.Private == 'No', 'Apps'].hist(label="Non Private College", bins=30)

plt.xlabel('Number of Applications')
plt.legend()
```

Out[11]: <matplotlib.legend.Legend at 0x2093c8deec8>



In [12]:

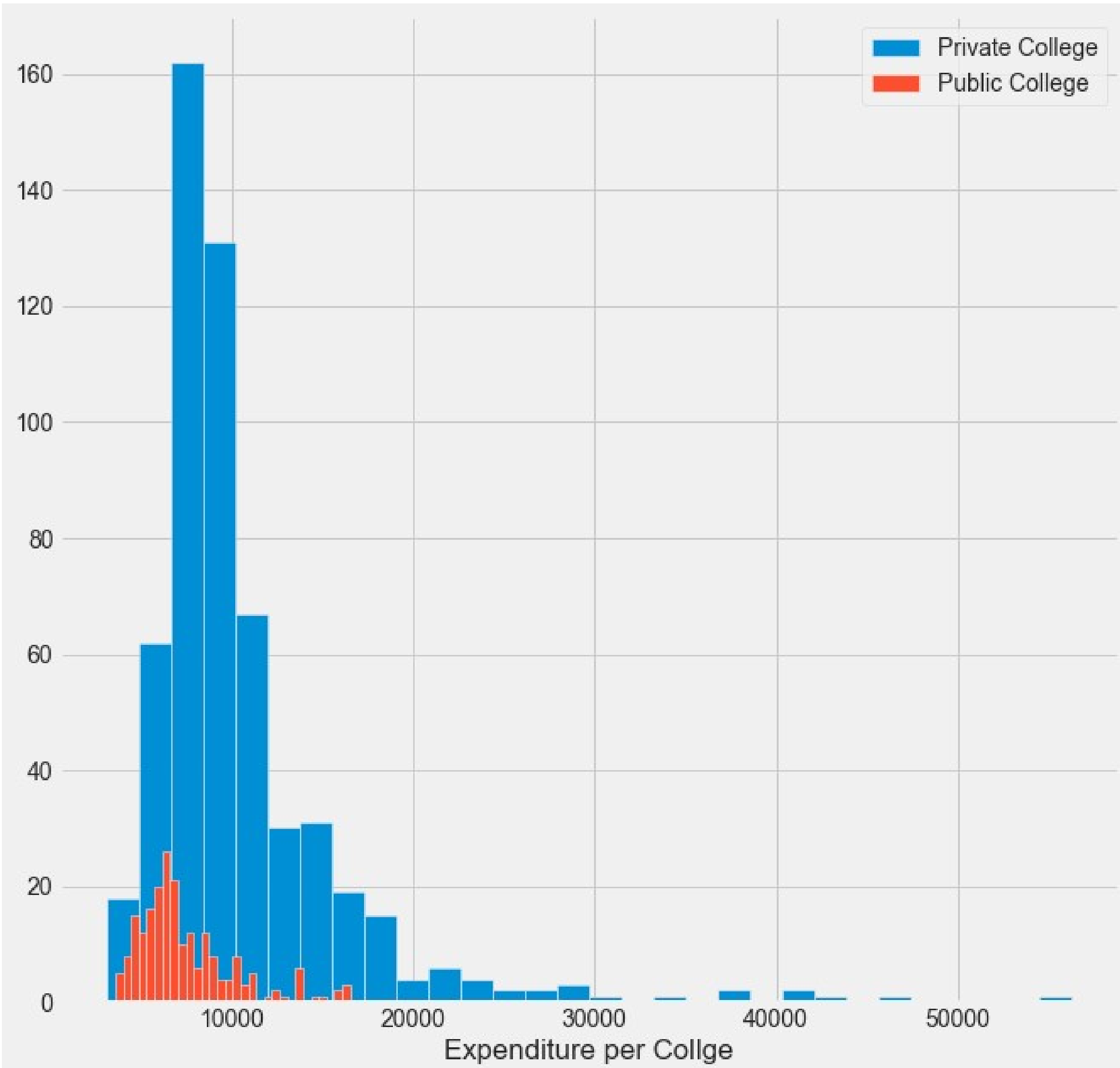
```
df.loc[df.Apps > 20000]
```

Out[12]:

	School	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F_Undergrad	P_Undergrad	Outstate	R
59	Boston University	Yes	20192	13007	3810	45	80	14971	3113	18420	
461	Purdue University at West Lafayette	No	21804	18744	5874	29	60	26213	4065	9556	
483	Rutgers at New Brunswick	No	48094	26330	4520	36	79	21401	3712	7410	

In [13]:
```
# Checking for outliers in 'Expend'
plt.figure(figsize=(10, 10))

df.loc[df.Private == 'Yes', 'Expend'].hist(label="Private College", bins=30)
df.loc[df.Private == 'No', 'Expend'].hist(label="Public College", bins=30)

plt.xlabel('Expenditure per Collge')
plt.legend()
```

Out[13]: <matplotlib.legend.Legend at 0x2093cca3508>



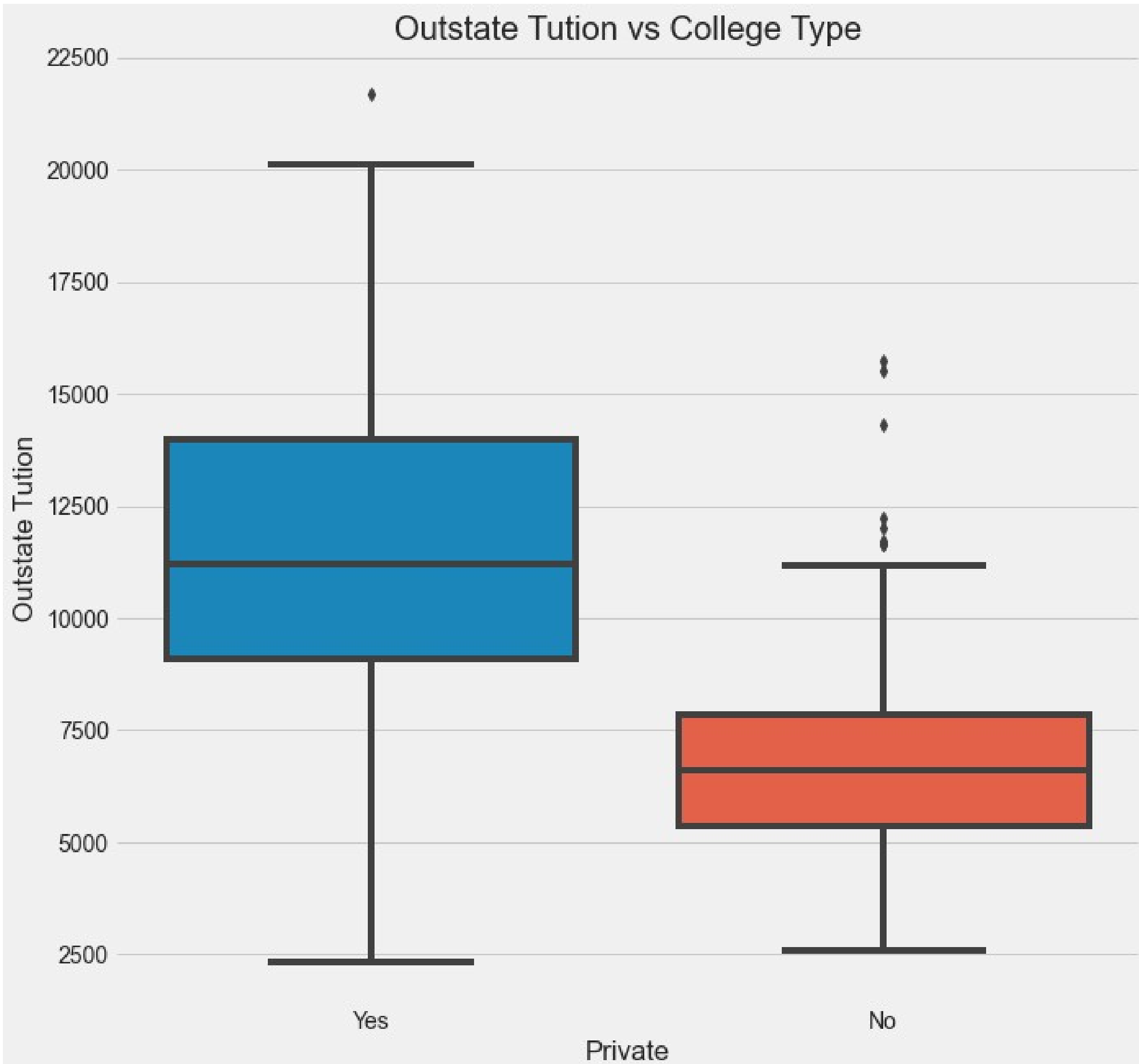
In [14]:
```
df.loc[df.Expend > 40000]
```

Out[14]:

	School	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F_Undergrad	P_Undergrad	Outstate
20	Antioch University	Yes	713	661	252	25	44	712	23	15476
284	Johns Hopkins University	Yes	8474	3446	911	75	94	3566	1569	18800
720	Wake Forest University	Yes	5661	2392	903	75	88	3499	172	13850
728	Washington University	Yes	7654	5259	1254	62	93	4879	1274	18350
775	Yale University	Yes	10705	2453	1317	95	99	5217	83	19840

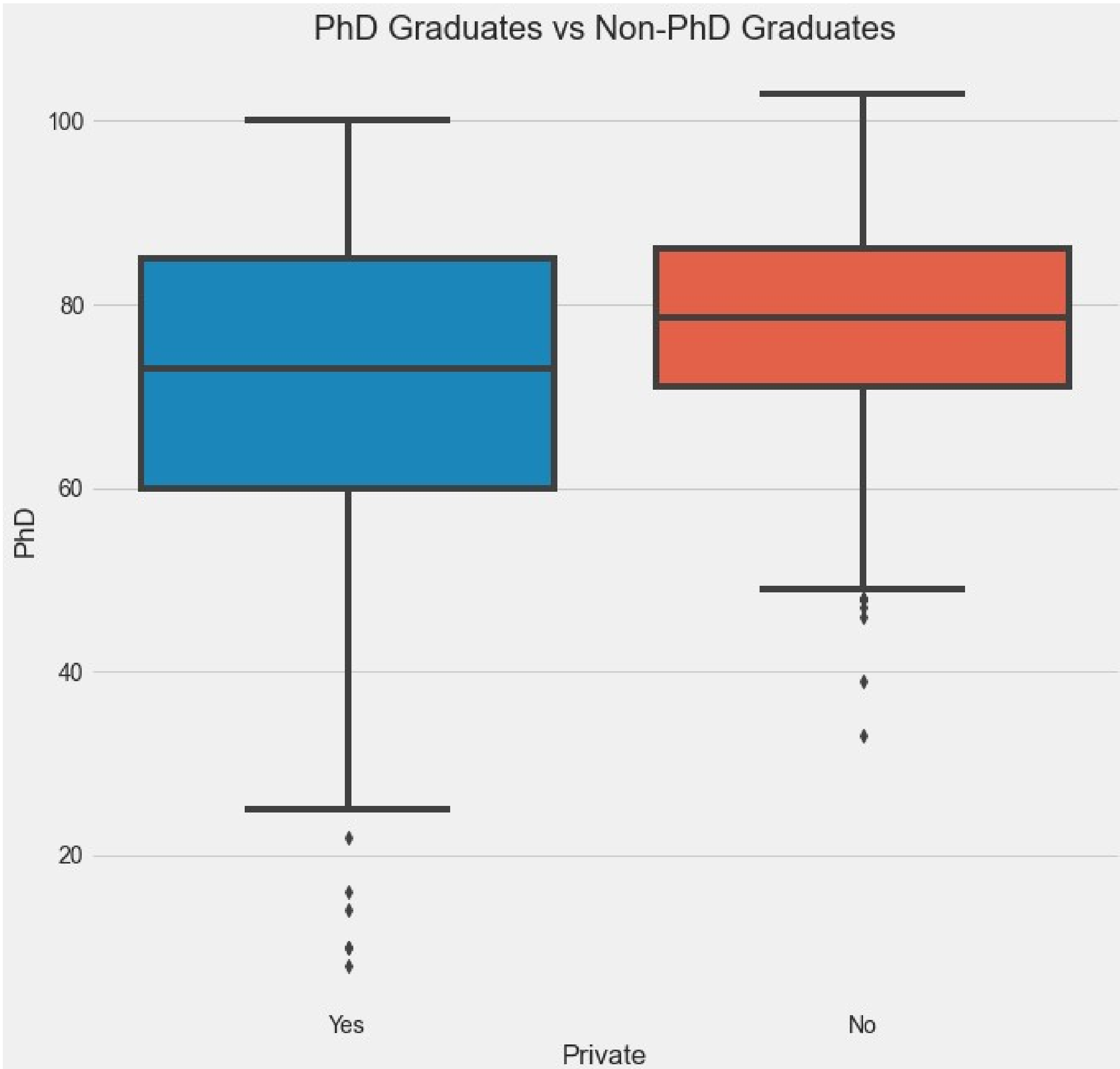
In [15]:

```
# Comparing 'out-of-state' tuition with Boxplots
import matplotlib.pyplot as plt
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111)

sns.boxplot(y="Outstate", x="Private", data=df)
ax.set_xlabel('Private')
ax.set_ylabel('Outstate Tution')
ax.set_title('Outstate Tution vs College Type')
plt.show()
```



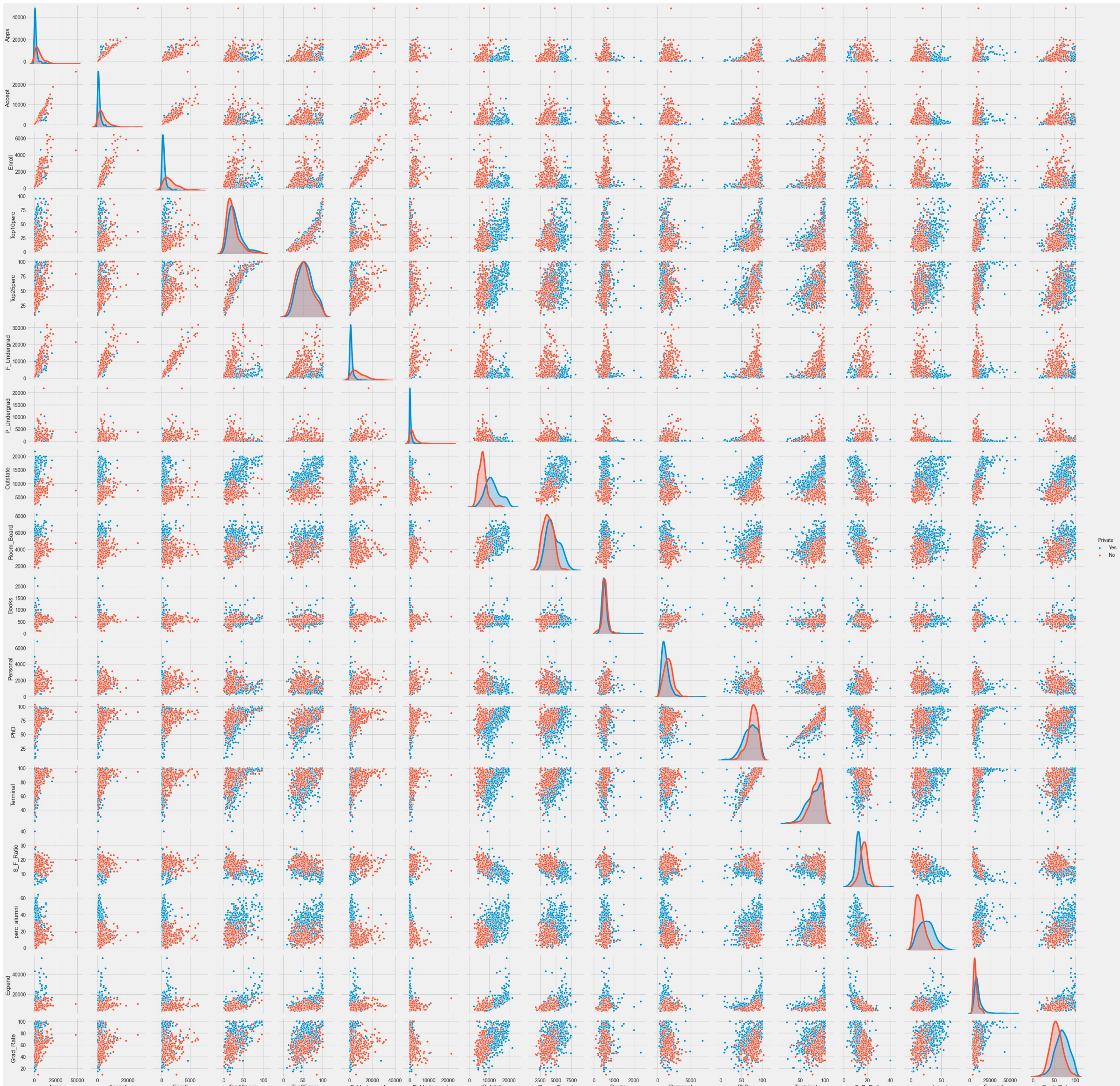
In [16]:

```
# # Comparing % faculty with 'PhD' with Boxplots
fig = plt.figure(figsize=(10, 10))
ax = fig.add_subplot(111)

sns.boxplot(x="Private", y="PhD", data=df)
ax.set_xlabel('Private')
ax.set_ylabel('PhD')
ax.set_title('PhD Graduates vs Non-PhD Graduates')
plt.show()
```



In [17]:
```
# Identifying correlations with the target variable
sns.pairplot(df, hue='Private')
```

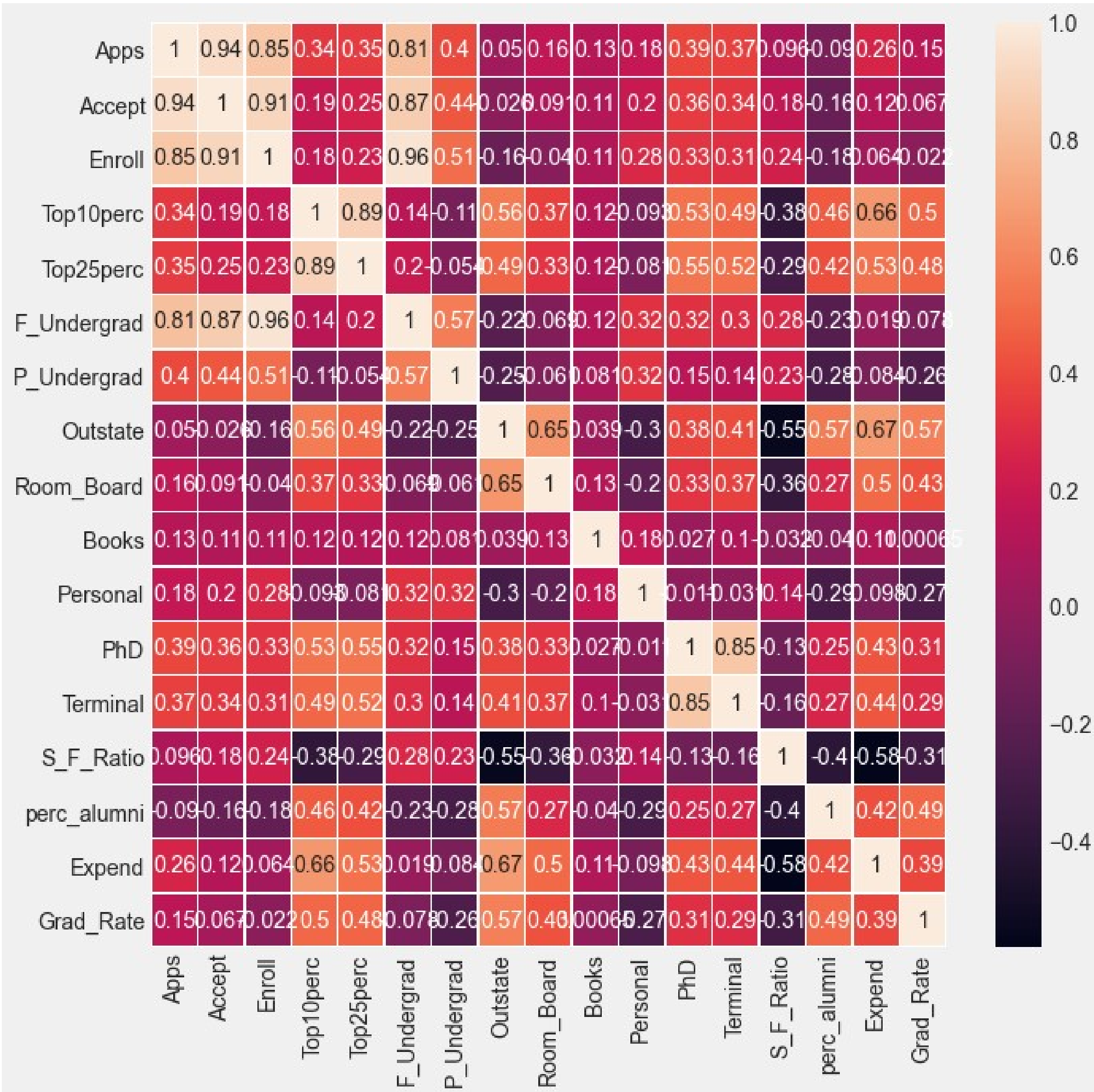
Out[17]: <seaborn.axisgrid.PairGrid at 0x2093cdb5d48>



In [18]:

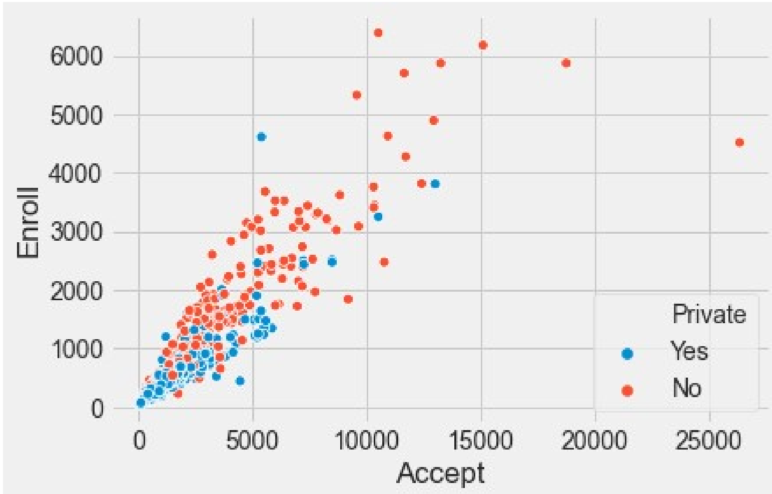
```python
# Identifying the features most related to the target variable
# using numerical values
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(df.corr(), annot=True, linewidth=.5, ax=ax)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x20947f01f48>



In [19]:
```
# Correlation between 'Accept' and 'Enroll'
sns.scatterplot(x=df["Accept"], y=df["Enroll"], data=df, hue="Private")
plt.figure(figsize=(12, 12))
```

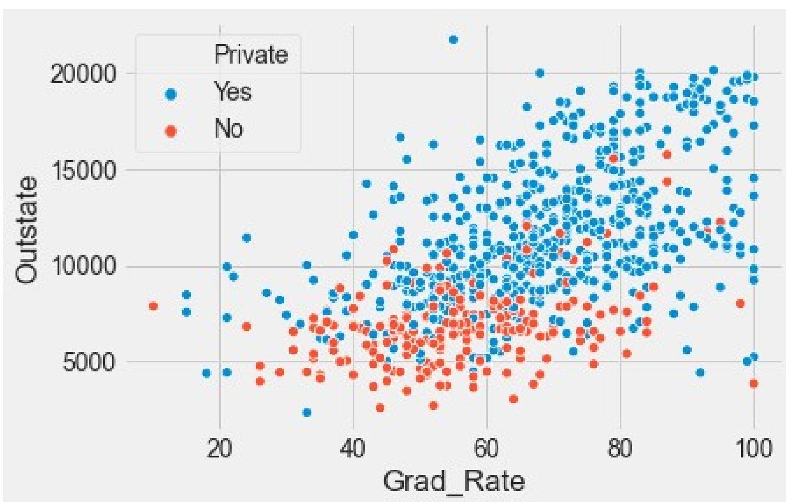
Out[19]: <Figure size 864x864 with 0 Axes>



<Figure size 864x864 with 0 Axes>

In [20]:
```
# Correlation between 'out-of-state tuition' and
# % of fuculty with 'Ph.D.'
sns.scatterplot(x=df["Grad_Rate"], y=df["Outstate"], data=df, hue="Private")
plt.figure(figsize=(12, 12))
```

Out[20]: <Figure size 864x864 with 0 Axes>



<Figure size 864x864 with 0 Axes>

In [21]:
```
# Changing the target variable from string to numerical data
y=df['Private'].astype('category').cat.codes
```

In [22]:
```
print(y)
```

```
0      1
1      1
2      1
3      1
4      1
      ..
772    0
773    1
774    1
775    1
776    1
Length: 777, dtype: int8
```

In [23]:
```
df.head()
```

Out[23]:

	School	Private	Apps	Accept	Enroll	Top10perc	Top25perc	F_Undergrad	P_Undergrad	Outstate	Roor
0	Abilene Christian University	Yes	1660	1232	721	23	52	2885	537	7440	
1	Adelphi University	Yes	2186	1924	512	16	29	2683	1227	12280	
2	Adrian College	Yes	1428	1097	336	22	50	1036	99	11250	
3	Agnes Scott College	Yes	417	349	137	60	89	510	63	12960	
4	Alaska Pacific University	Yes	193	146	55	16	44	249	869	7560	

In [24]:
```
df.Private.value_counts()
```

Out[24]:
```
Yes    565
No     212
Name: Private, dtype: int64
```

In [30]:
```
# Standardizing the input features
scaler = StandardScaler()
X = scaler.fit_transform(X)
X
```

Out[30]:
```
array([[-3.46881819e-01, -3.21205453e-01, -6.35089011e-02, ...,
        -8.67574189e-01, -5.01910084e-01, -3.18251941e-01],
       [-2.10884040e-01, -3.87029908e-02, -2.88584214e-01, ...,
        -5.44572203e-01,  1.66109850e-01, -5.51261842e-01],
       [-4.06865631e-01, -3.76317928e-01, -4.78121319e-01, ...,
         5.85934748e-01, -1.77289956e-01, -6.67766793e-01],
       ...,
       [-2.33895071e-01, -4.23771558e-02, -9.15087008e-02, ...,
        -2.21570217e-01, -2.56241250e-01, -9.59029170e-01],
       [ 1.99171118e+00,  1.77256262e-01,  5.78332661e-01, ...,
         2.12019418e+00,  5.88797079e+00,  1.95359460e+00],
       [-3.26765760e-03, -6.68715889e-02, -9.58163623e-02, ...,
         4.24433755e-01, -9.87115613e-01,  1.95359460e+00]])
```

In [45]:
```
# Split input features and target variable into training and testing data sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_sta
te=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.2,
random_state=1)
```

In [46]:
```
# Use a Sequential model and use a dense library to build 'input', 'hidden' and 'ou
tput' layers
model=Sequential()
model.add(Dense(60, input_dim=17, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

In [47]:
```
# Compile the Neural Network, we use 'Adam' to optimize our neural network
model.compile(loss='binary_crossentropy',
optimizer='adam',
metrics=['accuracy'])
```

In [59]:
```
# Fit the training data to the model, using a batch_size of 20, iterating over 20 E
pochs
seqModel=model.fit(X_train, y_train, epochs=90, batch_size=20, validation_data=(X_v
al, y_val))
```

```
Train on 434 samples, validate on 109 samples
Epoch 1/90
434/434 [==============================] - 0s 108us/step - loss: 0.0033 - accura
cy: 1.0000 - val_loss: 0.1755 - val_accuracy: 0.9450
Epoch 2/90
434/434 [==============================] - 0s 123us/step - loss: 0.0033 - accura
cy: 1.0000 - val_loss: 0.1759 - val_accuracy: 0.9450
Epoch 3/90
434/434 [==============================] - 0s 108us/step - loss: 0.0033 - accura
cy: 1.0000 - val_loss: 0.1772 - val_accuracy: 0.9358
Epoch 4/90
434/434 [==============================] - 0s 87us/step - loss: 0.0032 - accurac
y: 1.0000 - val_loss: 0.1786 - val_accuracy: 0.9450
Epoch 5/90
434/434 [==============================] - 0s 108us/step - loss: 0.0032 - accura
cy: 1.0000 - val_loss: 0.1754 - val_accuracy: 0.9450
Epoch 6/90
434/434 [==============================] - 0s 108us/step - loss: 0.0031 - accura
cy: 1.0000 - val_loss: 0.1780 - val_accuracy: 0.9450
Epoch 7/90
434/434 [==============================] - 0s 108us/step - loss: 0.0031 - accura
cy: 1.0000 - val_loss: 0.1770 - val_accuracy: 0.9450
Epoch 8/90
434/434 [==============================] - 0s 72us/step - loss: 0.0030 - accurac
y: 1.0000 - val_loss: 0.1771 - val_accuracy: 0.9450
Epoch 9/90
434/434 [==============================] - 0s 87us/step - loss: 0.0030 - accurac
y: 1.0000 - val_loss: 0.1799 - val_accuracy: 0.9450
Epoch 10/90
434/434 [==============================] - 0s 108us/step - loss: 0.0030 - accura
cy: 1.0000 - val_loss: 0.1749 - val_accuracy: 0.9450
Epoch 11/90
434/434 [==============================] - 0s 118us/step - loss: 0.0029 - accura
cy: 1.0000 - val_loss: 0.1832 - val_accuracy: 0.9358
Epoch 12/90
434/434 [==============================] - 0s 77us/step - loss: 0.0030 - accurac
y: 1.0000 - val_loss: 0.1781 - val_accuracy: 0.9450
Epoch 13/90
434/434 [==============================] - 0s 80us/step - loss: 0.0029 - accurac
y: 1.0000 - val_loss: 0.1819 - val_accuracy: 0.9450
Epoch 14/90
434/434 [==============================] - 0s 72us/step - loss: 0.0028 - accurac
y: 1.0000 - val_loss: 0.1801 - val_accuracy: 0.9450
Epoch 15/90
434/434 [==============================] - 0s 108us/step - loss: 0.0028 - accura
cy: 1.0000 - val_loss: 0.1822 - val_accuracy: 0.9450
Epoch 16/90
434/434 [==============================] - 0s 72us/step - loss: 0.0027 - accurac
y: 1.0000 - val_loss: 0.1818 - val_accuracy: 0.9450
Epoch 17/90
434/434 [==============================] - 0s 108us/step - loss: 0.0028 - accura
cy: 1.0000 - val_loss: 0.1743 - val_accuracy: 0.9450
Epoch 18/90
434/434 [==============================] - 0s 123us/step - loss: 0.0027 - accura
cy: 1.0000 - val_loss: 0.1837 - val_accuracy: 0.9358
Epoch 19/90
434/434 [==============================] - 0s 108us/step - loss: 0.0027 - accura
cy: 1.0000 - val_loss: 0.1794 - val_accuracy: 0.9450
Epoch 20/90
434/434 [==============================] - 0s 87us/step - loss: 0.0026 - accurac
y: 1.0000 - val_loss: 0.1822 - val_accuracy: 0.9450
Epoch 21/90
434/434 [==============================] - 0s 110us/step - loss: 0.0026 - accura
cy: 1.0000 - val_loss: 0.1809 - val_accuracy: 0.9358
```

In [56]:
```
# Evaluate the model
eval_model=model.evaluate(X_train, y_train)
eval_model
```

```
434/434 [==============================] - 0s 0us/step
```

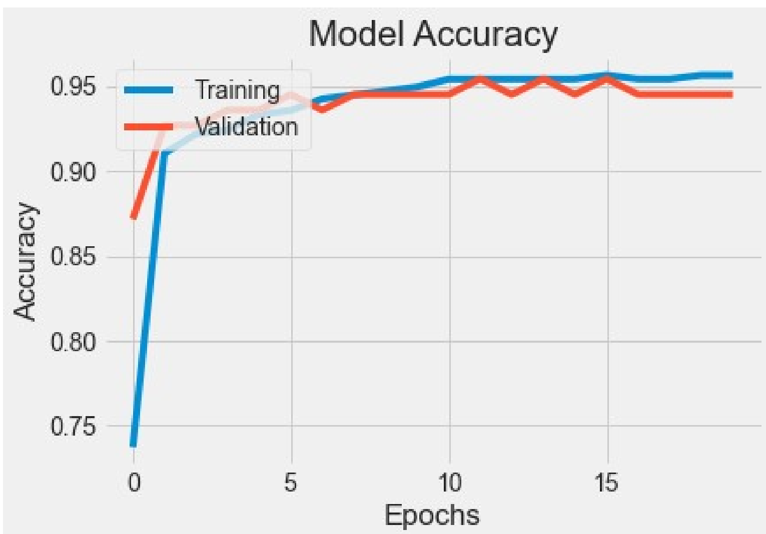
Out[56]:
```
[0.0032517095321729З, 1.0]
```

In [57]:
```
# Predict the output for the test dataset
y_pred=model.predict(X_test)
y_pred =(y_pred>0.5)
```

In [58]:
```
# Check the accuracy on the test dataset with a confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[ 58   6]
 [ 11 159]]
```

In [39]:
```
# Plot model accuracy
plt.plot(seqModel.history['accuracy'])
plt.plot(seqModel.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Training','Validation'],loc='upper left')
plt.show()
```



In [40]:
```python
# Plot model loss
plt.plot(seqModel.history['loss'])
plt.plot(seqModel.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['Training', 'Validation'], loc='upper left')
plt.show()
```

In [43]:
```python
# Save the model and architecture to a file
model.save('model.h5')
print('Save model to disk')
```

```
Save model to disk
```

Summary

The classification report shows the precision and recall of our NN model, the result shown is close to 100% accuracy. The Confusion Matrix is showing us the True positive and True negative result is 218 out of a toal 234 observations, in other words the accuracy for the test dataset is 92, but here it'a rounded to 1.0. Overall the prediction was highly accurate.

We've not been very selective about our choice of features and as a result, positive correlation between the test data and training data could cause overfitting problems. One way the model could be improved is by adding a number of regularization methods.

Accuracy & Loss: towards the end the 'Training Accuracy' is slightly higher than 'Validation Accuracy' which is ok, the same thing with 'Training Loss' which is slightly lower than 'Validation Loss'. Training the model initially with only 50 Epochs, the lines were meeting halfway and towards the end respectively, which is a sign of overfitting and which improved with increasing the epochs to 100.

Therefore as we train for more epochs the gaps should get wider with TL > VL and TA > VA.

In [ ]: