This code is for plotting #1 -- Part B

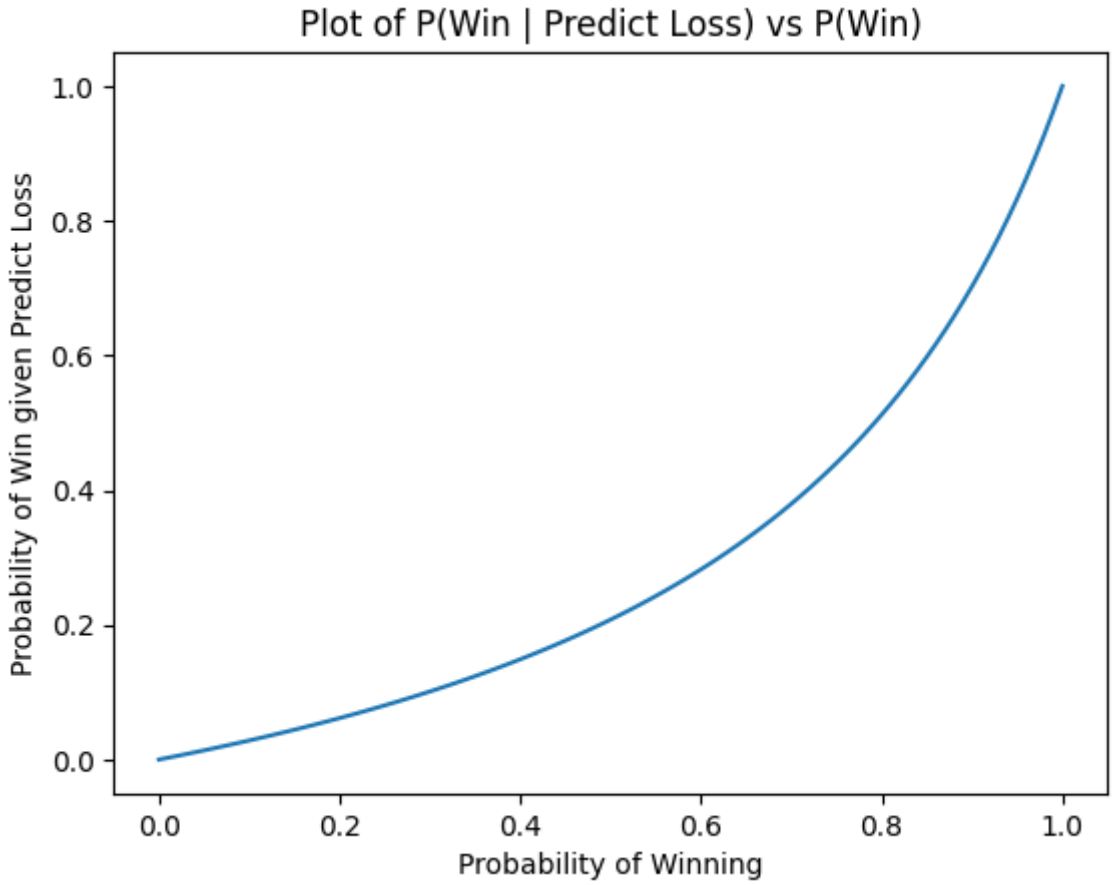
In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
#Create an array from 0 to 1 with 100 points
P_win = np.linspace(0,1,101)
P_win_given_predictloss = np.zeros(P_win.shape)
```

In [3]:
```
#Do Calculations
P_win_given_predictloss = (P_win * 0.234) / (0.892 - 0.658*P_win)
```

In [4]:
```
#Plotting
fig1 = plt.figure(num = 1, clear = True)
ax1 = fig1.add_subplot(1,1,1)
ax1.plot(P_win, P_win_given_predictloss)
ax1.set_xlabel("Probability of Winning")
ax1.set_ylabel("Probability of Win given Predict Loss")
ax1.set_title("Plot of P(Win | Predict Loss) vs P(Win)")

fig1.savefig("Problem 1 B.png")
```



In [6]:
```
print(P_win[np.where(np.abs(P_win_given_predictloss - 0.5) < 0.01)])
```

```
[0.79]
```

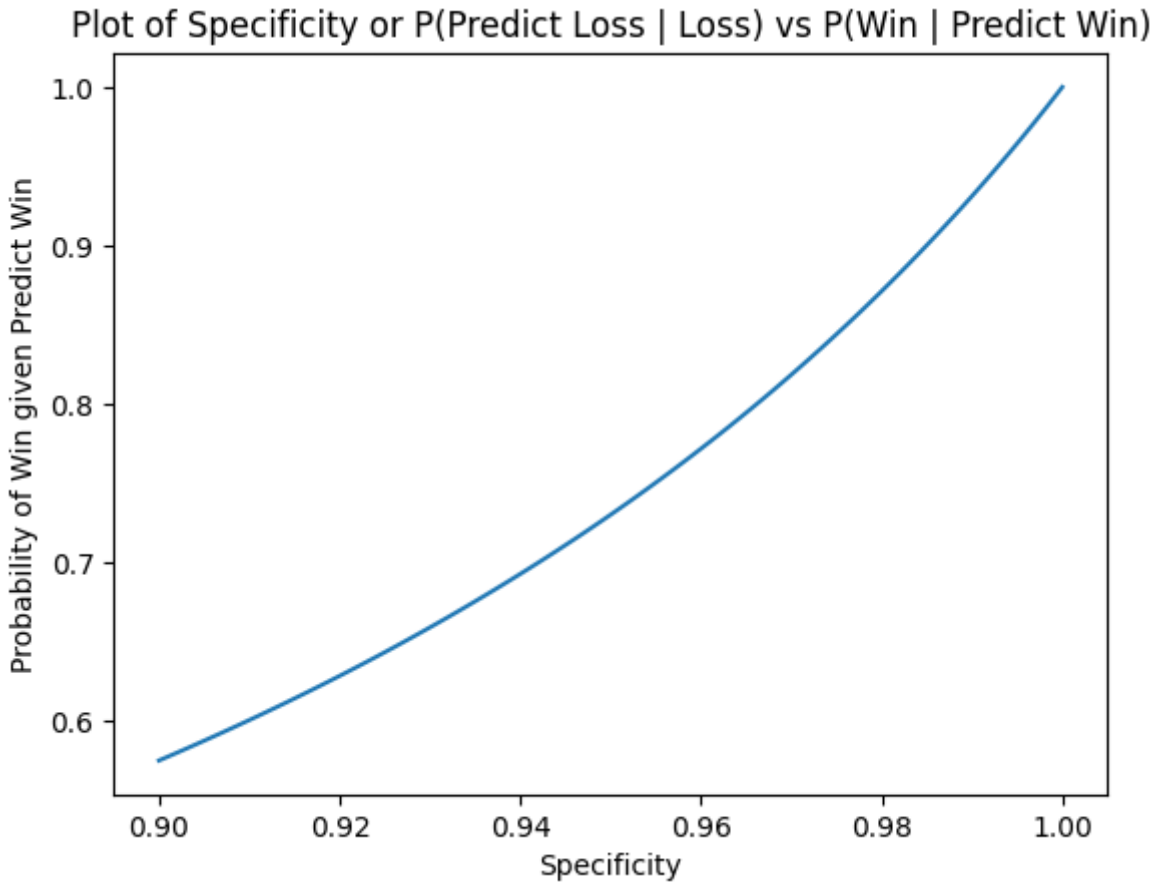
This code is for plotting #1 - Part E

In [7]:
```
#Create an array from 0.90 to 1.00 with 100 points
Specificity = np.linspace(0.9,1,101)
P_win_given_predictwin= np.zeros(Specificity.shape)
```

In [8]:
```
#Do Calculations
P_win_given_predictwin = 0.1149 /(0.9649 - 0.85 * Specificity)
```

In [9]:
```
#Plotting
fig2 = plt.figure(num = 2 , clear = True)
ax2 = fig2.add_subplot(1,1,1)
ax2.plot(Specificity, P_win_given_predictwin)
ax2.set_xlabel("Specificity")
ax2.set_ylabel("Probability of Win given Predict Win")
ax2.set_title("Plot of Specificity or P(Predict Loss | Loss) vs P(Win | Pre

fig2.savefig("Problem 1 E.png")
```



We can see a near linear curve in the above plot. It is interesting to note that just a small drop in specificity, such as from 100% specificity to 90% specificity, generates a large drop in the probability of winning given the model predicting a win.

Problem 1 - F

Accuracy is not a good measure. It is around 87.31%, which seems great, but given that P(win) = 15% only, predicting a loss every single time gives an accuracy of 85% already. Some version of AUC would be better.