ISC 4241 Activity 3 Problem 3

Data Used: “Microsoft*Results.CSV” with three variables: ID, HasDetect (observed value), and P* HasDetect (Model Predicted Probability).

Problem 2.1 (0 Points) Read the CSV file “Microsoft_Results.CSV”

In [1]:
```
import pandas as pd;

microsoft = pd.read_csv('C:/Users/danma/Downloads/Microsoft_Results.csv')
microsoft = microsoft.dropna()

microsoft.head()
```

Out[1]:

	MachineIdentifier	HasDetections	P_HasDetections
0	3fcee248758b525e5f2c018e64e431ee	0	0.626452
1	31f06c5a35708a7c3cd42123c4d94515	0	0.239536
2	c20ca6ece05a514b1057299e50d58aa2	0	0.504254
3	ea24238010ff3a76d7db824eeae24a8c	1	0.623444
4	ecdaa2783928b10a6036bbd39c95e7ad	1	0.546973

Problem 2.2 (3 Points) Write a program to calculate the following statistics: “True Positive”, “False Positive”, “True Negative”, “False Negative”, “Sensitivity”, “Specificity”, “Accuracy”, and “Precision” for any given cut-off probability (i.e., “P_HasDetect”). The input data set has three variables and the output data set has nine variables including “Cut off Probability”, “True Positive”, “False Positive”, “True Negative”, “False Negative”, “Sensitivity”, “Specificity”, “Accuracy”, and “Precision”.

Using 0.55 at Cutoff Probability for confusion matrix and etc.

In [2]:
```
import numpy as np;

microsoft['HasDetections'] = microsoft['HasDetections'].astype(float)

from sklearn.metrics import confusion_matrix;
y = microsoft['HasDetections']
y_pred = np.where(microsoft['P_HasDetections'] > 0.55, 1, 0)
y_pred_proba = microsoft['P_HasDetections']
#y_pred = microsoft['P_HasDetections']
conf_matrix = confusion_matrix(microsoft['HasDetections'], y_pred)
print("Confusion Matrix:\n", conf_matrix)
#print(classification_report(microsoft['HasDetections'], y_pred))

tp = np.sum(np.logical_and(y_pred ==1, microsoft['HasDetections']==1))
```

```
fp = np.sum(np.logical_and(y_pred ==1, microsoft['HasDetections']==0))
tn = np.sum(np.logical_and(y_pred == 0, microsoft['HasDetections']==0))
fn = np.sum(np.logical_and(y_pred == 0, microsoft['HasDetections']== 1))
print('\nTrue Positive: ', tp)
print('False Positive: ', fp)
print('True Negative: ', tn)
print('False Negative: ', fn)

sensitivity1 = conf_matrix[0,0]/(conf_matrix[0,0]+conf_matrix[0,1])
print('\nSensitivity : ', sensitivity1 )

specificity1 = conf_matrix[1,1]/(conf_matrix[1,0]+conf_matrix[1,1])
print('Specificity : ', specificity1)

#accuracy_score(microsoft['HasDetections'],microsoft['P_HasDetections'])
print("\nAccuracy and Precision:\n")
from sklearn.metrics import accuracy_score, recall_score, precision_score, roc_auc_sco

pd.DataFrame(data=[accuracy_score(microsoft['HasDetections'], y_pred), recall_score(mi
                   precision_score(microsoft['HasDetections'], y_pred), roc_auc_score(
                   index=["accuracy", "recall", "precision", "roc_auc_score"])
```

```
Confusion Matrix:
 [[401355  99644]
 [273469 225532]]

True Positive:  225532
False Positive:  99644
True Negative:  401355
False Negative:  273469

Sensitivity :  0.8011093834518632
Specificity :  0.4519670301261921

Accuracy and Precision:
```

Out[2]:

	0
accuracy	0.626887
recall	0.451967
precision	0.693569
roc_auc_score	0.626538

Problem 2.3 (3 Points) Write a program to calculate the AUC and Gini of the model.

In [3]:

```
from sklearn.metrics import roc_auc_score;
# Calculating AUC and GINI of Model
auc = roc_auc_score(y, y_pred_proba)
gini = 2*(auc-0.5)

print('AUC: %.2f' % auc)
print('GINI: %.2f' % gini)
```

```
AUC: 0.69
GINI: 0.39
```

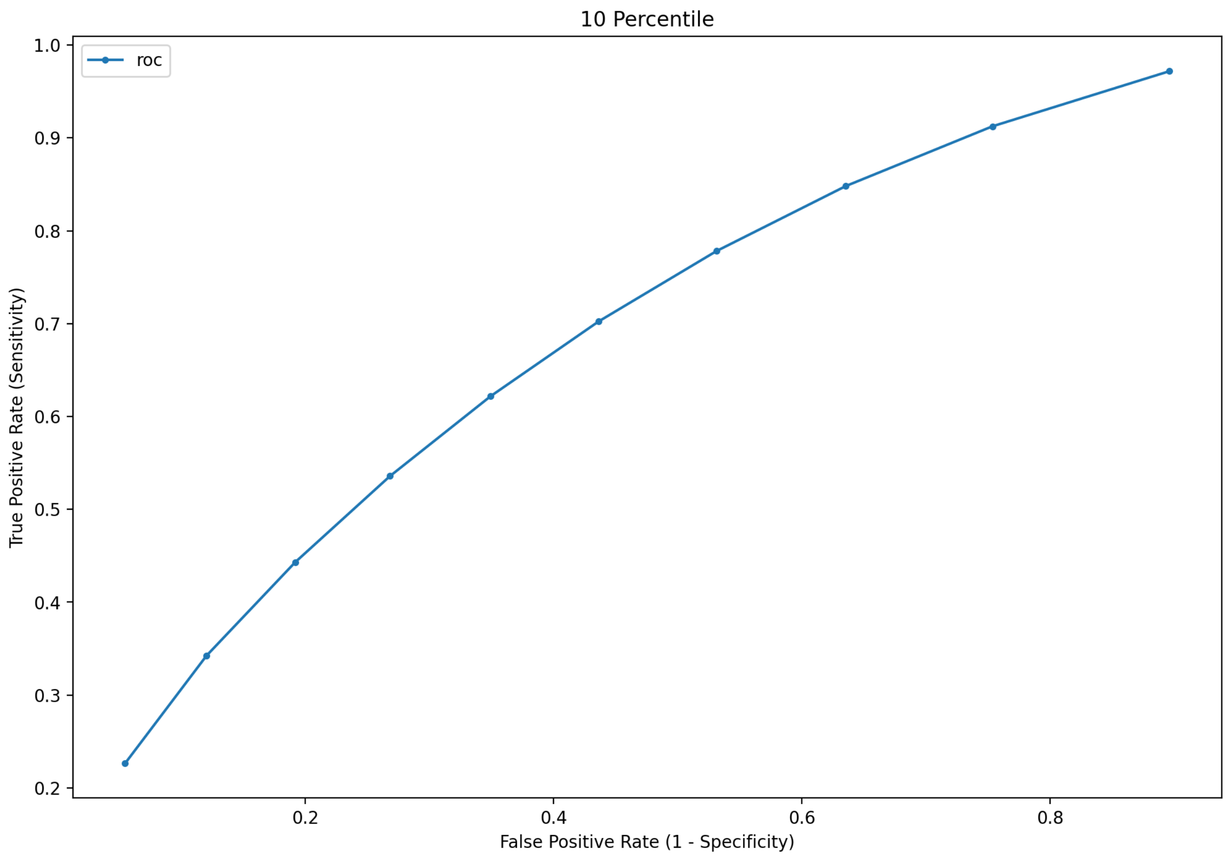
Problem 2.4 (3 Points) Write a program to produce the ROC curve of this model for each point on this curve being calculated at ten-percentile level (i.e., to calculate the (Sensitivity, 1-specificity) pair at cut-off probability of every ten percent).

In [4]:
```
import matplotlib.pyplot as plt;
from sklearn.metrics import roc_curve;
import numpy as np;
fpr, tpr, thresholds = roc_curve(y, y_pred_proba)

tpr10 = np.quantile(tpr, q = np.arange(0.095, 1, 0.095))
fpr10 = np.quantile(fpr, q = np.arange(0.095, 1, 0.095))
#shows 10 points
print(len(tpr10))

plt.figure(figsize=(12, 8), dpi=200)
plt.plot(fpr10, tpr10, label = 'roc',marker='.')
plt.legend(fontsize=16)
plt.title("10 Percentile")
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
# show the legend
plt.legend()
# show the plot
plt.show()
```

```
10
```



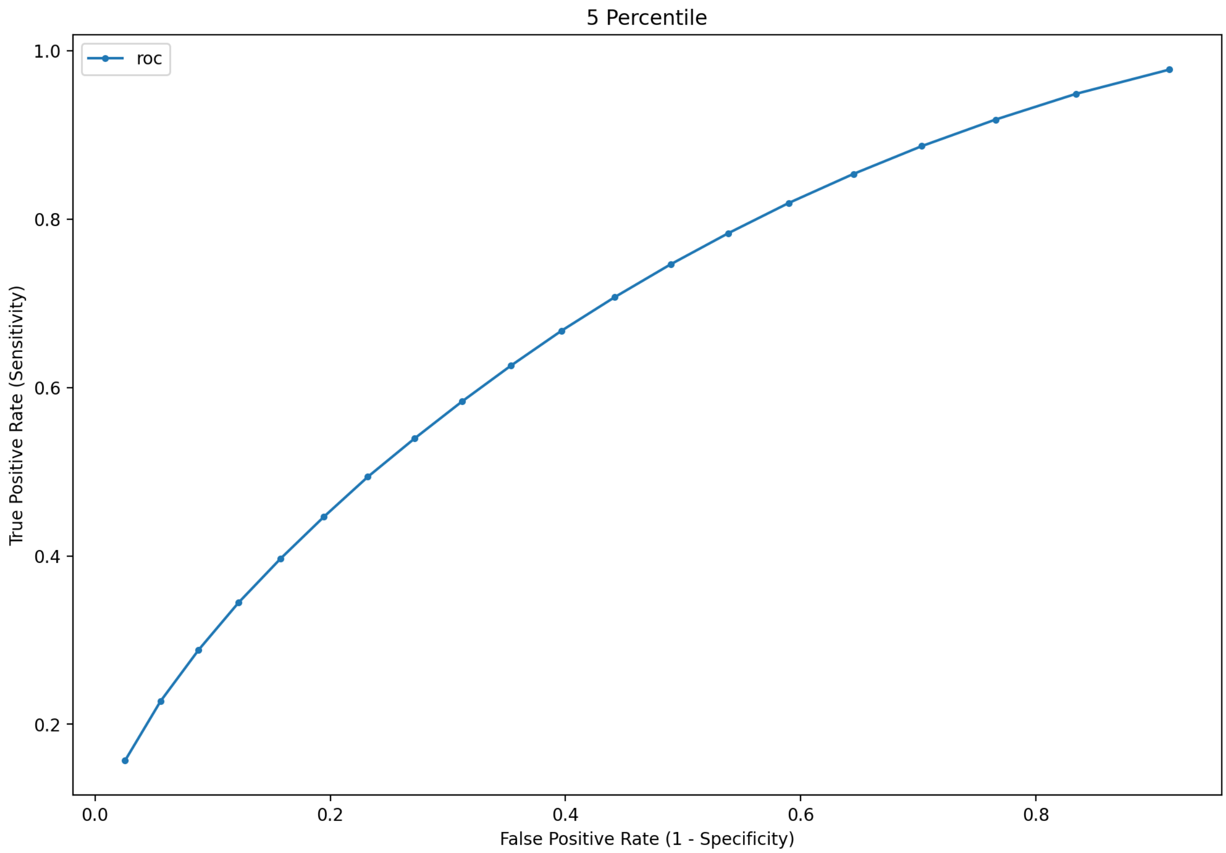
Problem 2.5 (3 Points) Write a program to produce the ROC curve of this model for each point on this curve being calculated

at five-percentile level (i.e., to calculate the (Sensitivity, 1-specificity) pair at cut-off probability of every five percent).

In [5]:
```
tpr5 = np.quantile(tpr, q = np.arange(0.048, 1, 0.048))
fpr5 = np.quantile(fpr, q = np.arange(0.048, 1, 0.048))
#shows 10 points
print(len(tpr5))

plt.figure(figsize=(12, 8), dpi=200)
plt.plot(fpr5, tpr5, label = 'roc',marker='.')
plt.legend(fontsize=16)
plt.title("5 Percentile")
plt.xlabel('False Positive Rate (1 - Specificity)')
plt.ylabel('True Positive Rate (Sensitivity)')
# show the legend
plt.legend()
# show the plot
plt.show()
```

```
20
```



Problem 2.6 (3 Points) Write a program to produce a lift chart of the model at decile level (i.e., every ten percent one point).

In [6]:
```
#divide each entry in decile by .55 cutoff
tpr10lift = tpr10 / .55
#get cummalative true positive rate
tpr10cumlift = np.cumsum(tpr10lift)

plt.figure(figsize=(12, 8), dpi=200)
plt.plot(fpr10, tpr10lift, label = 'Lift',marker='.')
```

```
plt.plot(fpr10, tpr10cumlift, label = 'Cummalative Lift',marker='.')
plt.legend(fontsize=16)
plt.title("Lift Chart")
plt.xlabel('Fraction')
plt.ylabel('Lift')
# show the legend
plt.legend()
# show the plot
plt.show()
```