In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, roc_auc_scor
e, roc_curve
import matplotlib.pyplot as plt
import numpy as np
```

In [4]:
```
data = pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

In [5]:
```
data.describe()
```

Out[5]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fracti
count	299.000000	299.000000	299.000000	299.000000	299.000000
mean	60.833893	0.431438	581.839465	0.418060	38.083612
std	11.894809	0.496107	970.287881	0.494067	11.834841
min	40.000000	0.000000	23.000000	0.000000	14.000000
25%	51.000000	0.000000	116.500000	0.000000	30.000000
50%	60.000000	0.000000	250.000000	0.000000	38.000000
75%	70.000000	1.000000	582.000000	1.000000	45.000000
max	95.000000	1.000000	7861.000000	1.000000	80.000000

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
age                         0
anaemia                     0
creatinine_phosphokinase    0
diabetes                    0
ejection_fraction           0
high_blood_pressure         0
platelets                   0
serum_creatinine            0
serum_sodium                0
sex                         0
smoking                     0
time                        0
DEATH_EVENT                 0
dtype: int64
```

In [7]:
```
data.corr()['DEATH_EVENT'].sort_values()
```

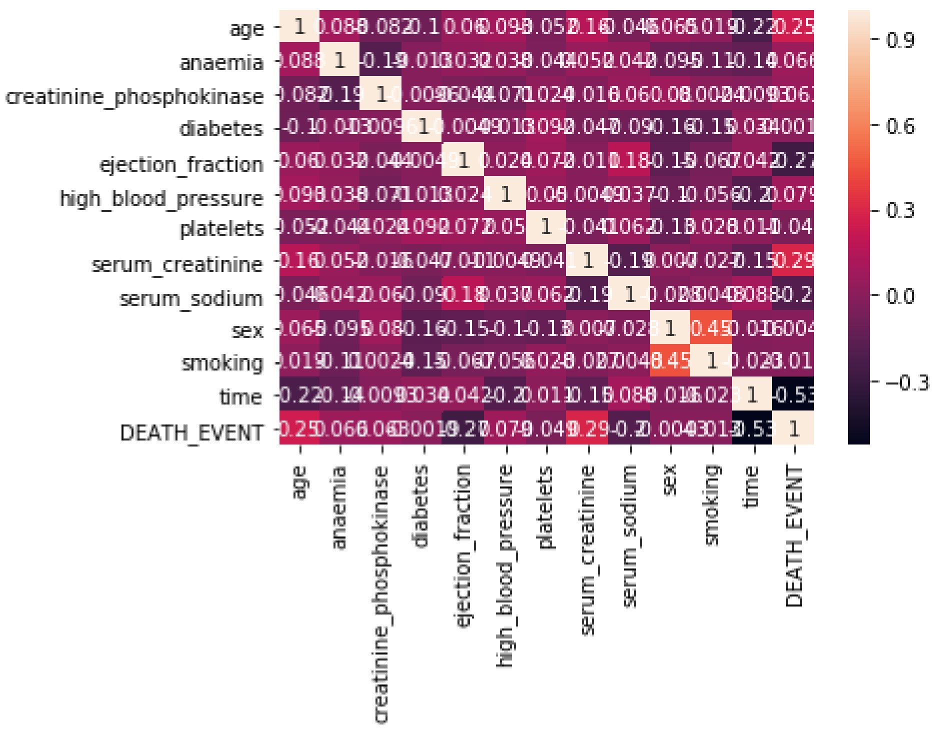
Out[7]:
```
time                       -0.526964
ejection_fraction          -0.268603
serum_sodium               -0.195204
platelets                  -0.049139
smoking                    -0.012623
sex                        -0.004316
diabetes                   -0.001943
creatinine_phosphokinase    0.062728
anaemia                     0.066270
high_blood_pressure         0.079351
age                         0.253729
serum_creatinine            0.294278
DEATH_EVENT                 1.000000
Name: DEATH_EVENT, dtype: float64
```

In [8]:
```
plt.figure(figsize=(12, 6))
```

Out[8]:
```
<Figure size 864x432 with 0 Axes>

<Figure size 864x432 with 0 Axes>
```

In [9]:
```
sns.heatmap(data.corr(), annot=True)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d7b23a36a0>
```



In [10]:
```
X = data.drop('DEATH_EVENT', axis=1)
y = data['DEATH_EVENT']
```

In [11]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, ra
ndom_state=29)
```

In [12]:
```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
```

In [13]:
```
# Train the classifier using cross-validation
cross_val_scores = cross_val_score(rf_classifier, X_train, y_train, cv=5, s
coring='accuracy')
mean_cross_val_accuracy = cross_val_scores.mean()
```

In [14]:
```
# Train the classifier on the full training set
rf_classifier.fit(X_train, y_train)
```

Out[14]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=1,
            oob_score=False, random_state=42, verbose=0, warm_start=False)
```

In [15]:
```
# Make predictions on the test set
predictions = rf_classifier.predict(X_test)
```

In [17]:
```
# Evaluate and print the accuracy
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy (Before Cross-Validation): {accuracy:.4f}")
```

```
Accuracy (Before Cross-Validation): 0.8500
```

In [18]:
```
# Display Cross-Validation Mean Accuracy
print(f"Cross-Validation Mean Accuracy: {mean_cross_val_accuracy:.4f}")
```

```
Cross-Validation Mean Accuracy: 0.8326
```

In [19]:
```
# Confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Confusion Matrix:
[[144  23]
 [ 13  60]]
```

In [20]:
```
# AUC-ROC Curve
y_proba = rf_classifier.predict_proba(X_test)[:, 1]
roc_auc = roc_auc_score(y_test, y_proba)
fpr, tpr, _ = roc_curve(y_test, y_proba)
```

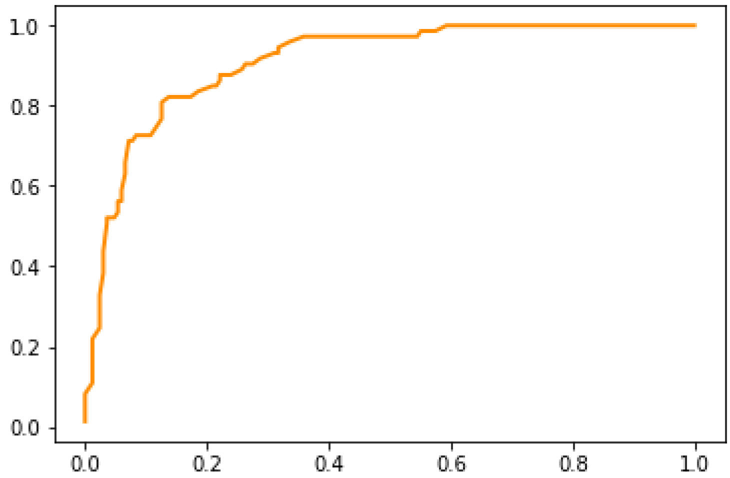
In [21]:
```
plt.figure(figsize=(8, 6))
```

Out[21]:
```
<Figure size 576x432 with 0 Axes>
```

```
<Figure size 576x432 with 0 Axes>
```

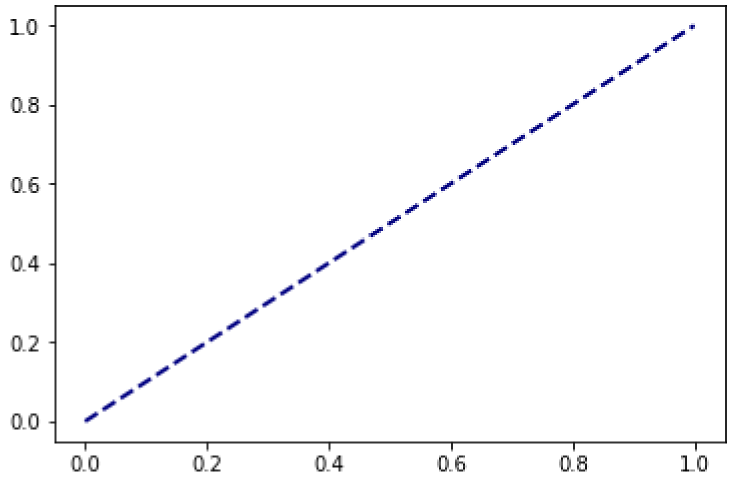
In [22]:
```
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = {:.2
f})'.format(roc_auc))
```

Out[22]: [<matplotlib.lines.Line2D at 0x1d7b276a8d0>]



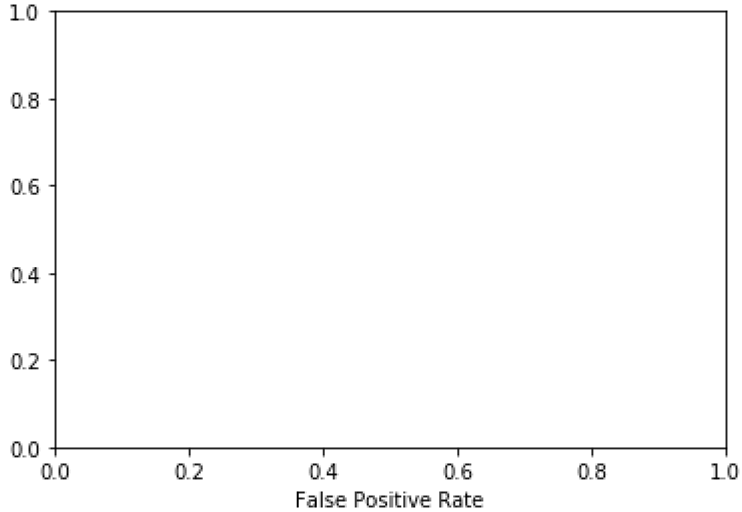
In [23]:
```
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
```

Out[23]: [<matplotlib.lines.Line2D at 0x1d7b27c60b8>]



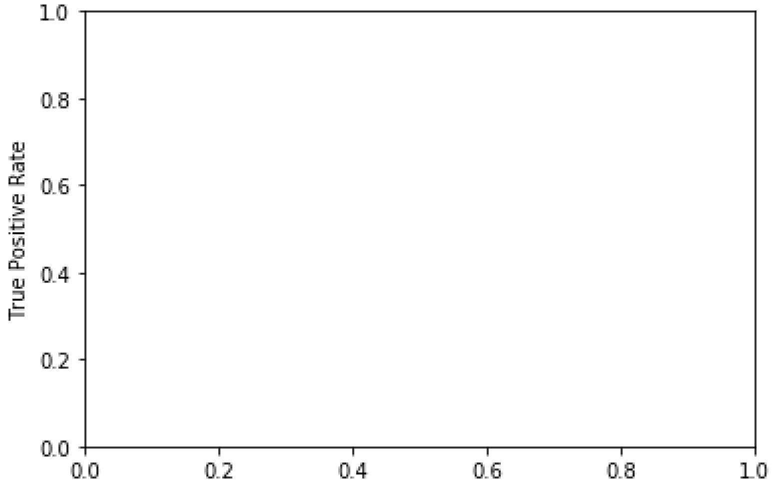
In [24]:
```
plt.xlabel('False Positive Rate')
```

Out[24]: Text(0.5,0,'False Positive Rate')



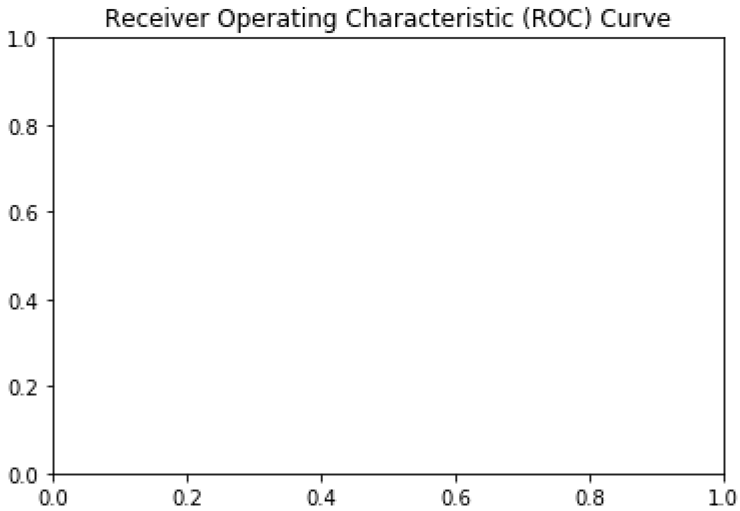
In [25]:
```
plt.ylabel('True Positive Rate')
```

Out[25]: `Text(0,0.5,'True Positive Rate')`



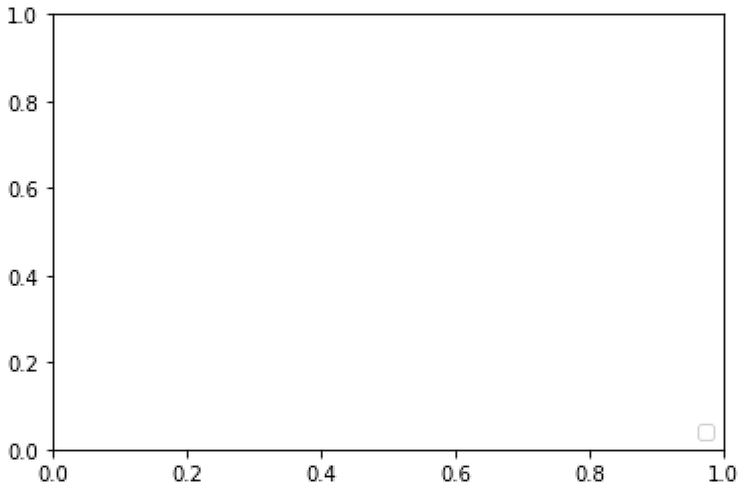
In [26]:
```
plt.title('Receiver Operating Characteristic (ROC) Curve')
```

Out[26]: `Text(0.5,1,'Receiver Operating Characteristic (ROC) Curve')`



In [29]:
```
plt.legend(loc='lower right')
plt.show()
```

No handles with labels found to put in legend.



In [32]:
```
from sklearn.metrics import log_loss
```

In [33]:
```
# Accuracy and Log Loss Curves during Training
train_acc = []
test_acc = []
train_logloss = []
test_logloss = []
```

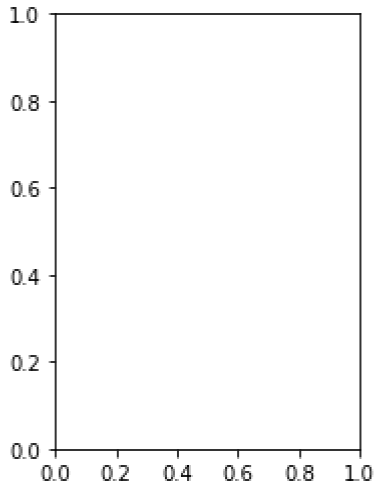
In [34]:
```
for n_estimators in range(1, 101):
    rf_classifier = RandomForestClassifier(n_estimators=n_estimators, rando
m_state=42)
    rf_classifier.fit(X_train, y_train)

    # Training Accuracy and Log Loss
    train_preds = rf_classifier.predict(X_train)
    train_accuracy = accuracy_score(y_train, train_preds)
    train_log_loss = log_loss(y_train, rf_classifier.predict_proba(X_trai
n))

    # Test Accuracy and Log Loss
    test_preds = rf_classifier.predict(X_test)
    test_accuracy = accuracy_score(y_test, test_preds)
    test_log_loss = log_loss(y_test, rf_classifier.predict_proba(X_test))

    # Append values
    train_acc.append(train_accuracy)
    test_acc.append(test_accuracy)
    train_logloss.append(train_log_loss)
    test_logloss.append(test_log_loss)
```

In [35]:
```
# Plotting Accuracy Curves
plt.figure(figsize=(12, 4))
```

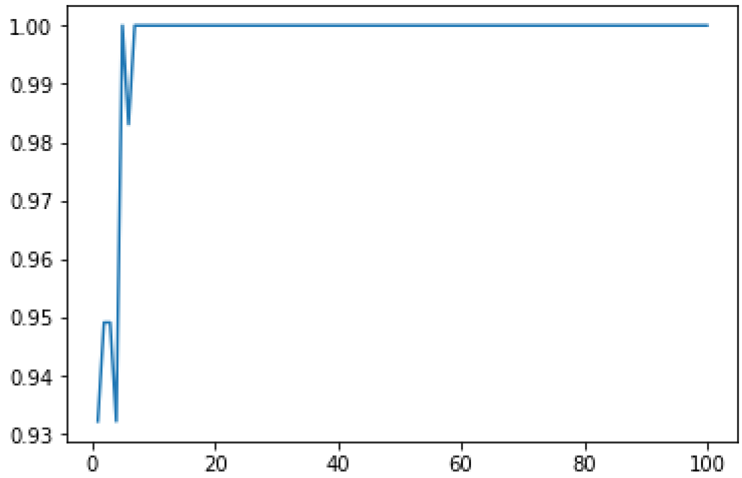
Out[35]:
```
<Figure size 864x288 with 0 Axes>
```
```
<Figure size 864x288 with 0 Axes>
```

In [36]:
```
plt.subplot(1, 2, 1)
```

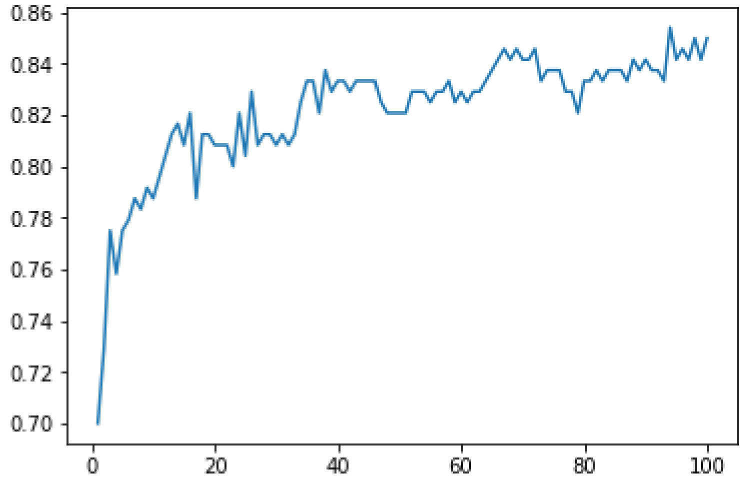
Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1d7b29a9780>
```



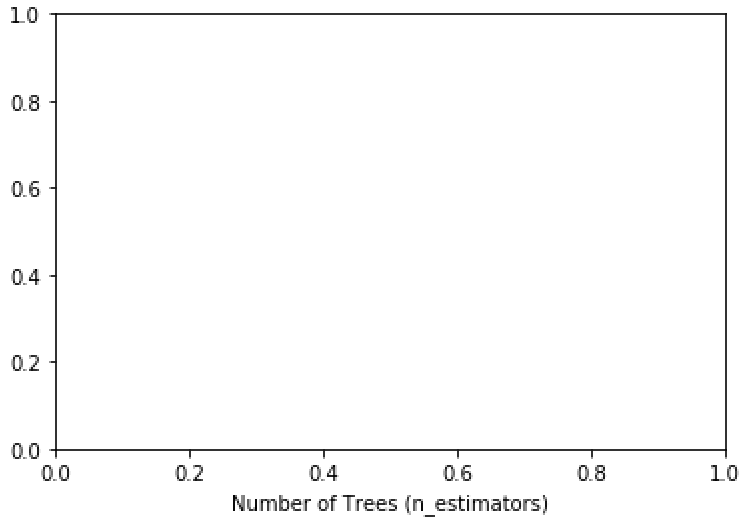
In [37]:
```
plt.plot(range(1, 101), train_acc, label='Training Accuracy')
```
Out[37]: [<matplotlib.lines.Line2D at 0x1d7b2a1cdd8>]



In [38]:
```
plt.plot(range(1, 101), test_acc, label='Testing Accuracy')
```
Out[38]: [<matplotlib.lines.Line2D at 0x1d7b2a7b908>]

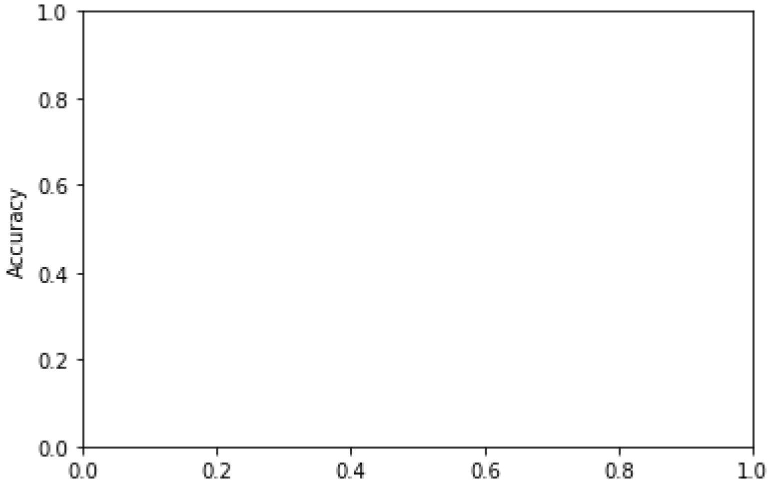


In [39]:
```
plt.xlabel('Number of Trees (n_estimators)')
```
Out[39]: Text(0.5,0,'Number of Trees (n_estimators)')



In [40]:
```
plt.ylabel('Accuracy')
```

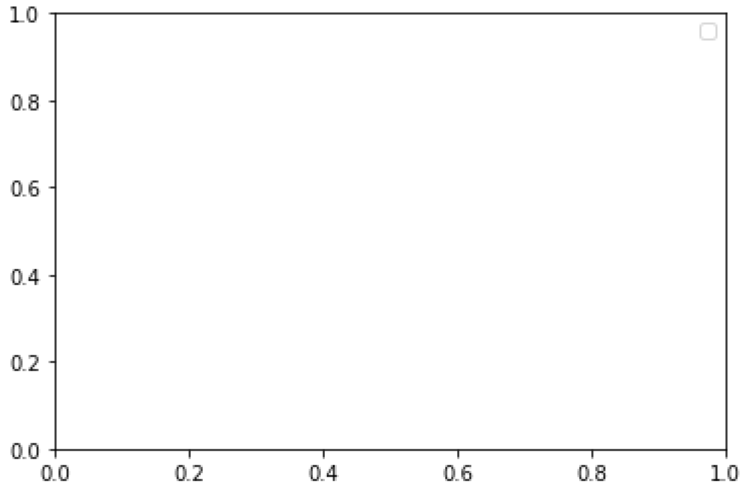
Out[40]: Text(0,0.5,'Accuracy')



In [41]:
```
plt.legend()
```

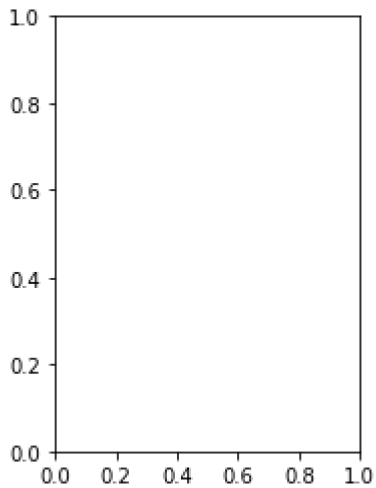
No handles with labels found to put in legend.

Out[41]: <matplotlib.legend.Legend at 0x1d7b2b592b0>



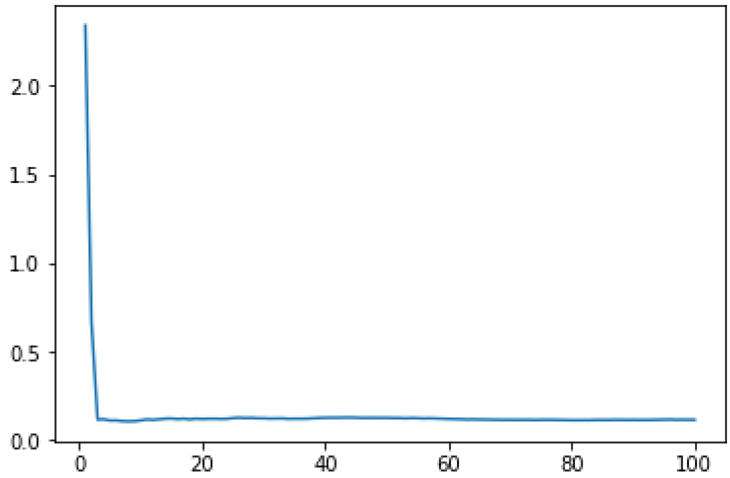
In [42]:
```
# Plotting Log Loss Curves
plt.subplot(1, 2, 2)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x1d7b3b6fa58>



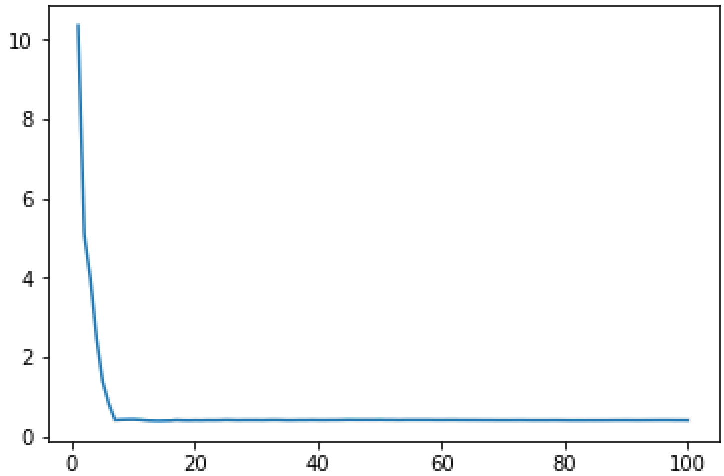
In [43]:
```
plt.plot(range(1, 101), train_logloss, label='Training Log Loss')
```

Out[43]: [<matplotlib.lines.Line2D at 0x1d7b3be2908>]



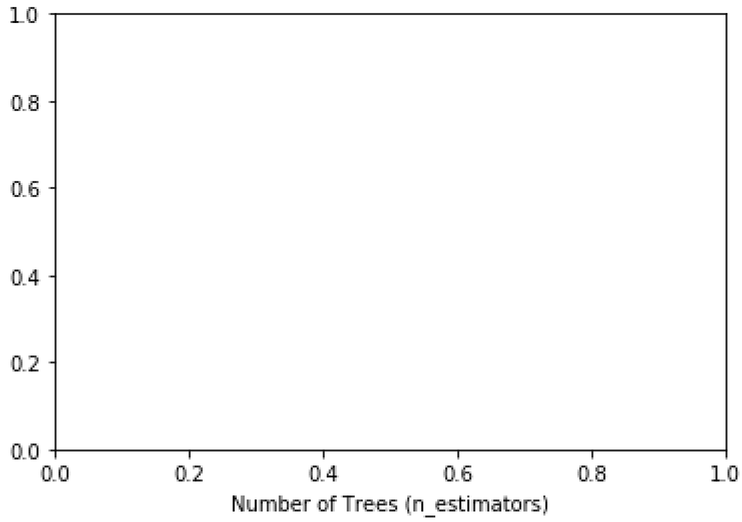
In [44]:
```
plt.plot(range(1, 101), test_logloss, label='Testing Log Loss')
```

Out[44]: [<matplotlib.lines.Line2D at 0x1d7b3c3b710>]



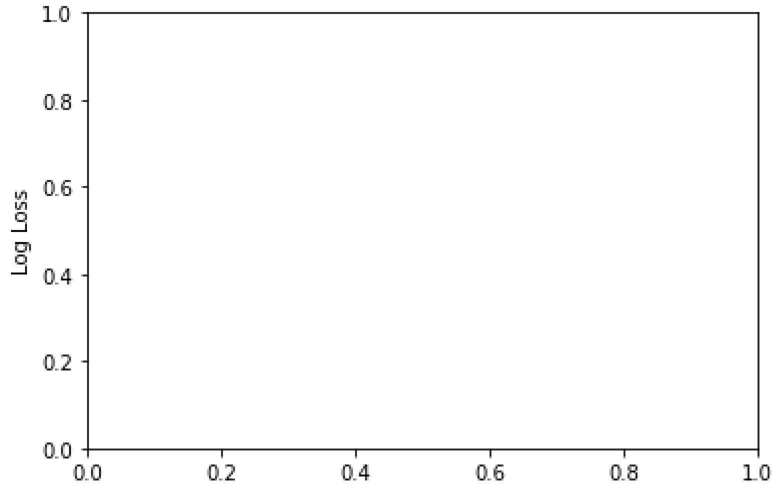
In [45]:
```
plt.xlabel('Number of Trees (n_estimators)')
```

Out[45]: Text(0.5,0,'Number of Trees (n_estimators)')



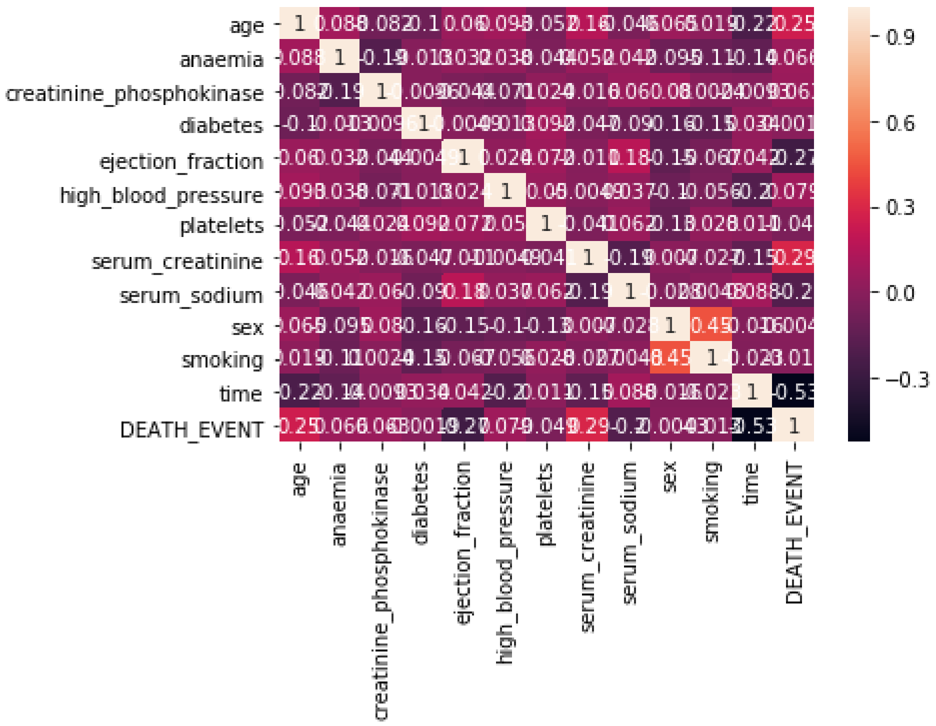
In [46]:
```
plt.ylabel('Log Loss')
```

Out[46]: Text(0,0.5,'Log Loss')



In [52]:
```
sns.heatmap(data.corr(), annot=True)
```

Out[52]: <matplotlib.axes._subplots.AxesSubplot at 0x1d7b3c14eb8>



In [53]:
```
accuracy = accuracy_score(y_test, predictions)
print(f"Accuracy (Before Cross-Validation): {accuracy:.4f}")
```

Accuracy (Before Cross-Validation): 0.8500

In [54]:
```
# Display Cross-Validation Mean Accuracy
print(f"Cross-Validation Mean Accuracy: {mean_cross_val_accuracy:.4f}")
```

Cross-Validation Mean Accuracy: 0.8326