Receive Operator Characteristic (ROC) curves

Summarize the predictive performance of a classification model at all classification thresholds. Specifically, plots the **true positive rate (TPR)** over the **false positive rate (FPR)**.

$$\text{True Positive Rate (TPR)} = \frac{TP}{TP + FN}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

In [1]:
```
#import libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, roc_auc_score
```

Synthetic data generation

In [2]:
```
#generate the synthetic data
X, y = make_classification(n_samples = 2000, n_classes = 2, n_features = 10, random
```

In [3]:
```
#add randomness to the data
random_state = np.random.RandomState(42)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]
```

Train and test set splitting

In [4]:
```
#split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_s
```

Classification models

1. Random forest classifier
2. Naive Bayes classifier
3. Logistic regression

Random forest

In [5]:
```
#build the classification model
rf = RandomForestClassifier(max_features = 5,
                            n_estimators = 100,
                            random_state = 42,
                            n_jobs = -1)
rf.fit(X_train, y_train)
```

Out[5]:
```
RandomForestClassifier(max_features=5, n_jobs=-1, random_state=42)
```

Naive Bayes

In [6]:
```
#build the classification model
nb = GaussianNB()
nb.fit(X_train, y_train)
```

Out[6]:
```
GaussianNB()
```

Logistic regression

In [7]:
```
#build the classification model
lr = LogisticRegression(fit_intercept = True)
lr.fit(X_train, y_train)
```

Out[7]:
```
LogisticRegression()
```

Classification results

Predictive probabilities

Calculate the predictive probability thresholds to test the balance and trade-off between TPR and FPR for each model.

In [8]:
```
#calculate the predictive probabilities for each model
r_probs = [0 for _ in range(len(y_test))]
rf_probs = rf.predict_proba(X_test)[:, 1]
nb_probs = nb.predict_proba(X_test)[:, 1]
lr_probs = lr.predict_proba(X_test)[:, 1]
```

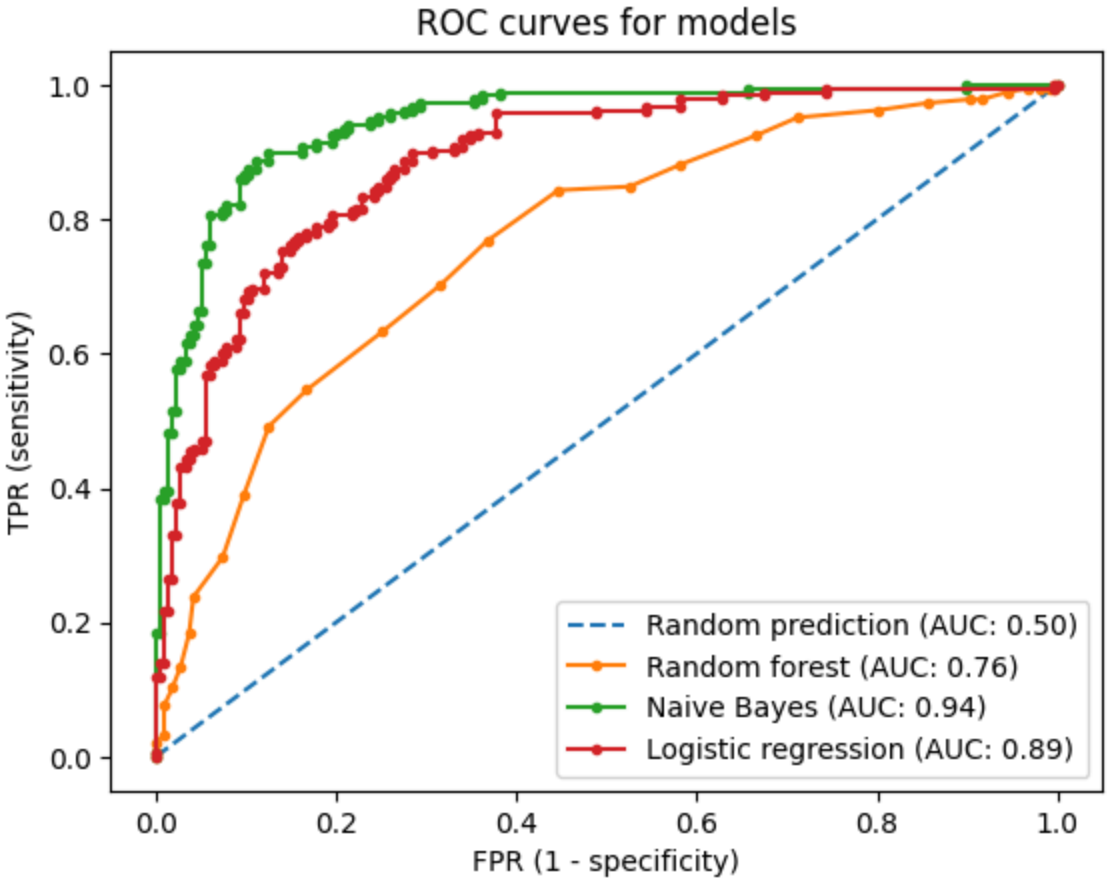
In [9]:
```
#calculate the ROC curves for each model
r_fpr, r_tpr, _ = roc_curve(y_test, r_probs)
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)
nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
lr_fpr, lr_tpr, _ = roc_curve(y_test, lr_probs)
```

AUC scores

In [10]:
```
#calculate the ROC AUC scores for each model
r_auc = roc_auc_score(y_test, r_probs)
rf_auc = roc_auc_score(y_test, rf_probs)
nb_auc = roc_auc_score(y_test, nb_probs)
lr_auc = roc_auc_score(y_test, lr_probs)
```

ROC plot

In [11]:
```
#plot the ROC curves for each model
plt.plot(r_fpr, r_tpr, linestyle = '--', label = f"Random prediction (AUC: {r_auc:.
plt.plot(rf_fpr, rf_tpr, marker = '.', label = f"Random forest (AUC: {rf_auc:.2f})"
plt.plot(nb_fpr, nb_tpr, marker = '.', label = f"Naive Bayes (AUC: {nb_auc:.2f})")
plt.plot(lr_fpr, lr_tpr, marker = '.', label = f"Logistic regression (AUC: {lr_auc:
plt.xlabel('FPR (1 - specificity)')
plt.ylabel('TPR (sensitivity)')
plt.title('ROC curves for models')
plt.legend()
plt.show()
```



Conclusion

The best performing classification model is the Gaussian Naive Bayes with the highest Area Under the ROC Cruve (AUC) of 0.94. This model has the highest true postive rate (TPR) across all false positive rates (FPR), providing it with more flexibility to balance sensitivity and specificity.