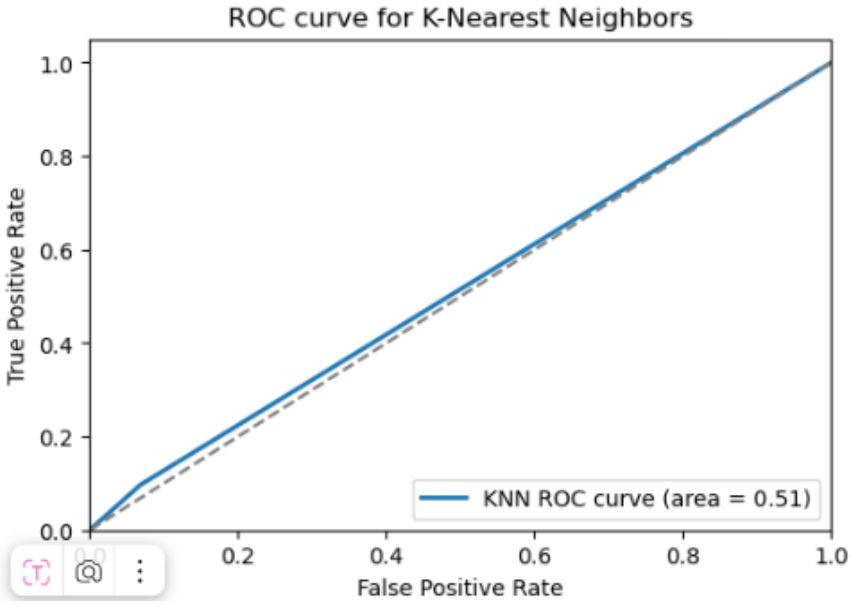
Составление более точных логов для домашнего задания модели обучений данных, Звягинцев 22П-2 –

```
# Список классификаторов
classifiers = ["KNeighborsClassifier", "Gaussian Naive Bayes", "Random Forest Classifier", "Logistic Regression"]

# Создание датафрейма для логов
log_cols = ["Classifier", "Accuracy", "Precision", "Recall", "F1 Score"]
log = pd.DataFrame(columns=log_cols)

# Заполнение данных для каждой модели
log["Classifier"] = classifiers

# Предполагается, что вы уже рассчитали точности
log["Accuracy"] = [knn_accuracy, gnb_accuracy, rfc_accuracy, lr_accuracy]

# Вычисление Precision, Recall и F1 Score для каждой модели
log["Precision"] = [
    precision_score(y_test, knn.predict(X_test)),
    precision_score(y_test, gnb.predict(X_test)),
    precision_score(y_test, rfc.predict(X_test)),
    precision_score(y_test, lr.predict(X_test1))
]

log["Recall"] = [
    recall_score(y_test, knn.predict(X_test)),
    recall_score(y_test, gnb.predict(X_test)),
    recall_score(y_test, rfc.predict(X_test)),
    recall_score(y_test, lr.predict(X_test1))
]

log["F1 Score"] = [
    f1_score(y_test, knn.predict(X_test)),
    f1_score(y_test, gnb.predict(X_test)),
    f1_score(y_test, rfc.predict(X_test)),
    f1_score(y_test, lr.predict(X_test1))
]

# Вывод результатов
print(log)
```

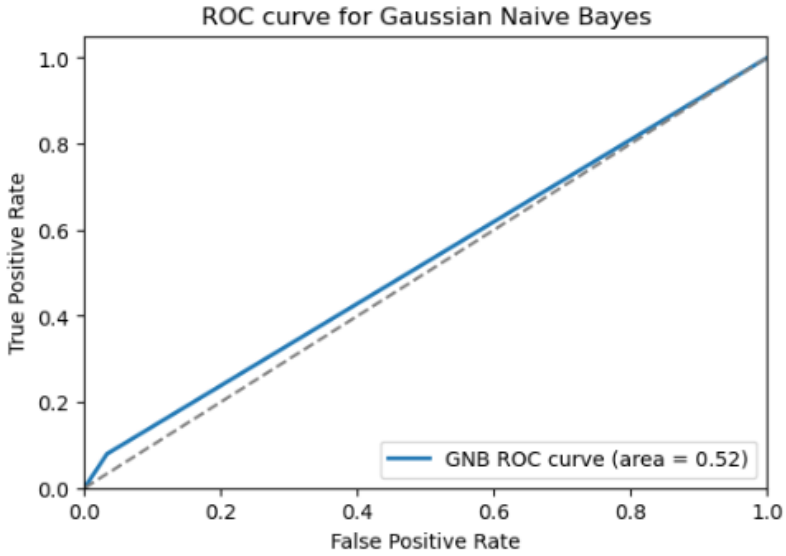
```
                  Classifier  Accuracy  Precision    Recall  F1 Score
0      KNeighborsClassifier  0.769091   0.255144  0.096423  0.139955
1      Gaussian Naive Bayes  0.793939   0.366906  0.079316  0.130435
2  Random Forest Classifier  0.871212   0.804469  0.447900  0.575425
3       Logistic Regression  0.811515   0.542857  0.206843  0.299550
```

[61]:
```python
from sklearn.metrics import RocCurveDisplay, roc_auc_score, roc_curve
import matplotlib.pyplot as plt
import seaborn as sns

# ROC-кривая для K-ближайших соседей
plt.figure(figsize=(6, 4))
fpr_knn, tpr_knn, thresholds_knn = roc_curve(y_test, y_test_predict)
roc_auc_knn = roc_auc_score(y_test, y_test_predict)

plt.plot(fpr_knn, tpr_knn, lw=2, label='KNN ROC curve (area = {:.2f})'.format(roc_auc_knn))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for K-Nearest Neighbors')
plt.legend(loc='lower right')
plt.savefig("ROC_KNN.png")
plt.show()
```

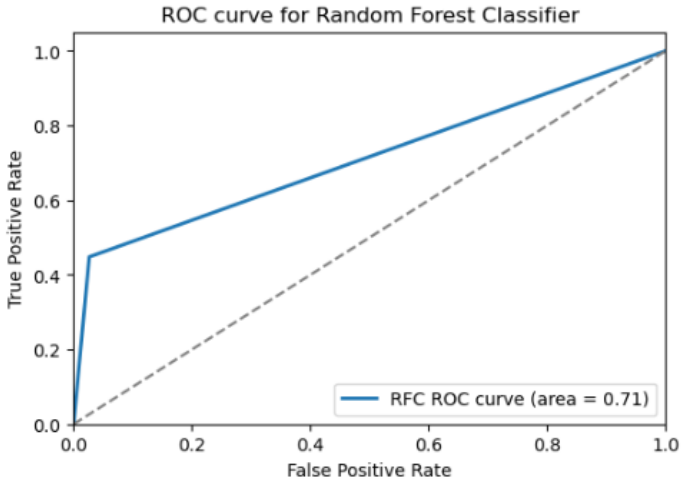


*[98]:
```
gnb_accuracy = accuracy_score(y_test, gnb_pred_test)
```

[132]:
```
# ROC-кривая для Наивного Байесовского Классификатора
plt.figure(figsize=(6, 4))
fpr_gnb, tpr_gnb, thresholds_gnb = roc_curve(y_test, gnb_pred_test)
roc_auc_gnb = roc_auc_score(y_test, gnb_pred_test)

plt.plot(fpr_gnb, tpr_gnb, lw=2, label='GNB ROC curve (area = {:.2f})'.format(roc_auc_gnb))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for Gaussian Naive Bayes')
plt.legend(loc='lower right')
plt.savefig("ROC_GNB.png")
plt.show()
```



[87]:
```
#roc_curve() вычисляет значения ложных положительных и истинных положительных показателей.
#roc_auc_score() вычисляет AUC для оценки качества классификации.
#Мы создаем график ROC-кривой с помощью matplotlib.
#На графике показана кривая с ложными положительными значениями по оси X и истинными положительными значениями по оси Y.
#Площадь под кривой (AUC) также отображается в легенде.
# Прогнозирование на тестовых данных для Random Forest
rfc_pred_test = rfc.predict(X_test)

# ROC-кривая для Классификатора Случайного Леса
plt.figure(figsize=(6, 4))
fpr_rfc, tpr_rfc, thresholds_rfc = roc_curve(y_test, rfc_pred_test)
roc_auc_rfc = roc_auc_score(y_test, rfc_pred_test)

plt.plot(fpr_rfc, tpr_rfc, lw=2, label='RFC ROC curve (area = {:.2f})'.format(roc_auc_rfc))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for Random Forest Classifier')
plt.legend(loc='lower right')
plt.savefig("ROC_RFC.png")
plt.show()
```



*[107]:
```
# ROC-кривая для Логистической Регрессии
plt.figure(figsize=(6, 4))
fpr_lr, tpr_lr, thresholds_lr = roc_curve(y_test, lr_pred_test)
roc_auc_lr = roc_auc_score(y_test, lr_pred_test)

plt.plot(fpr_lr, tpr_lr, lw=2, label='LR ROC curve (area = {:.2f})'.format(roc_auc_lr))
plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve for Logistic Regression')
plt.legend(loc='lower right')
plt.savefig("ROC_LR.png")
plt.show()
```