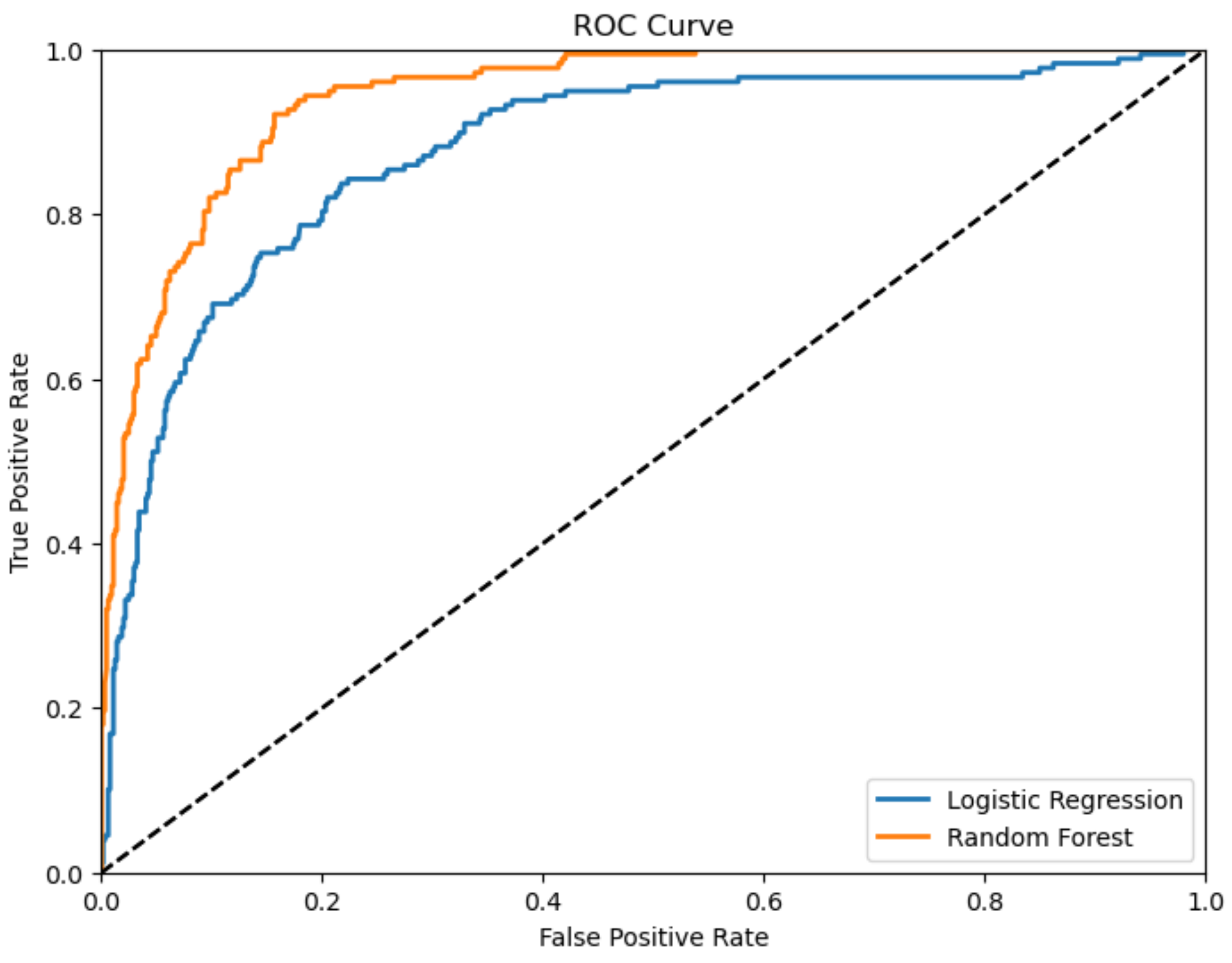
```
In [3]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import matplotlib.backends.backend_pdf
        from sklearn.model_selection import train_test_split, GridSearchCV
        from sklearn.preprocessing import StandardScaler
        from sklearn.linear_model import LogisticRegression
        from sklearn.ensemble import RandomForestClassifier
        from sklearn.metrics import roc_auc_score, roc_curve
        from sklearn.utils import resample


        # Загрузка данных
        data = pd.read_csv('grant_data_imb.csv')

        # Подготовка данных
        X = data.drop('Grant.Status', axis=1)
        y = data['Grant.Status']

        # Заполнение пропусков в числовых признаках
        num_features = X.select_dtypes(include=['int64', 'float64']).columns
        X[num_features] = X[num_features].fillna(X[num_features].mean())

        # Заполнение пропусков в категориальных признаках
        cat_features = X.select_dtypes(include=['object']).columns
        X[cat_features] = X[cat_features].fillna('Missing')

        # Прямое кодирование для категориальных признаков
        X = pd.get_dummies(X, columns=cat_features)

        # Разделение данных на обучающую и тестовую выборки
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,

        # Масштабирование признаков
        scaler = StandardScaler()
        X_train_scaled = scaler.fit_transform(X_train)
        X_test_scaled = scaler.transform(X_test)

        # Балансировка классов методом oversampling
        X_train_scaled_upsampled, y_train_upsampled = resample(
            X_train_scaled[y_train == 1],
            y_train[y_train == 1],
            replace=True,
            n_samples=X_train_scaled[y_train == 0].shape[0],
            random_state=123
        )

        X_train_balanced = np.vstack((X_train_scaled[y_train == 0], X_train_scale
        y_train_balanced = np.hstack((y_train[y_train == 0], y_train_upsampled))

        # Обучение модели логистической регрессии
        log_reg = LogisticRegression(max_iter=1000)
        log_reg.fit(X_train_balanced, y_train_balanced)
        y_pred_log_reg = log_reg.predict_proba(X_test_scaled)[:, 1]
```

```python
# Обучение модели случайного леса с подбором гиперпараметров
rf = RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20],
    'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(rf, param_grid, cv=5, scoring='roc_auc')
grid_search.fit(X_train_balanced, y_train_balanced)

best_rf = grid_search.best_estimator_
y_pred_rf = best_rf.predict_proba(X_test_scaled)[:, 1]

# Вывод результатов
print('ROC AUC for Logistic Regression:', roc_auc_score(y_test, y_pred_lo
print('Best parameters for Random Forest:', grid_search.best_params_)
print('ROC AUC for Random Forest:', roc_auc_score(y_test, y_pred_rf))

# Построение и сохранение ROC-кривых в PDF
with matplotlib.backends.backend_pdf.PdfPages("roc_curves.pdf") as pdf:
    plt.figure(figsize=(8, 6))
    plot_roc_curve(y_test, y_pred_log_reg, 'Logistic Regression')
    plot_roc_curve(y_test, y_pred_rf, 'Random Forest')
    plt.legend()
    plt.tight_layout()
    pdf.savefig()  # Сохраняем текущую фигуру в pdf-файл
    plt.close()

print("ROC-кривые успешно сохранены в 'roc_curves.pdf'")

# Функция для построения ROC-кривой
def plot_roc_curve(y_true, y_scores, label=None):
    fpr, tpr, thresholds = roc_curve(y_true, y_scores)
    plt.plot(fpr, tpr, linewidth=2, label=label)
    plt.plot([0, 1], [0, 1], 'k--') # dashed diagonal
    plt.axis([0, 1, 0, 1])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC Curve')

# Построение ROC-кривых
plt.figure(figsize=(8, 6))
plot_roc_curve(y_test, y_pred_log_reg, 'Logistic Regression')
plot_roc_curve(y_test, y_pred_rf, 'Random Forest')
plt.legend()
plt.show()
```

```
ROC AUC for Logistic Regression: 0.877567285079697
Best parameters for Random Forest: {'max_depth': 20, 'min_samples_split':
2, 'n_estimators': 200}
ROC AUC for Random Forest: 0.9435284382893476
ROC-кривые успешно сохранены в 'roc_curves.pdf'
```


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression
Random Forest

```
In [ ]:
```