```
import pandas as pd
df = pd.read_csv("/content/Illness.csv")
df.head()
```

	time	trt	age	wtkg	hemo	homo	drugs	karnof	oprior	z30	...	str2	strat	symptom	treat	offtrt	cd40	cd420	cd80	cd820
0	948	2	48	89.8128	0	0	0	100	0	0	...	0	1	0	1	0	422	477	566	324
1	1002	3	61	49.4424	0	0	0	90	0	1	...	1	3	0	1	0	162	218	392	564
2	961	3	45	88.4520	0	1	1	90	0	1	...	1	3	0	1	1	326	274	2063	1893
3	1166	3	47	85.2768	0	1	0	100	0	1	...	1	3	0	1	0	287	394	1590	966
4	1090	0	43	66.6792	0	1	0	100	0	1	...	1	3	0	0	0	504	353	870	782

5 rows × 23 columns

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, f1_score
import matplotlib.pyplot as plt
```

+ Code + Text

```
# Assuming you have a DataFrame called 'df' with the independent and dependent variables
X = df.drop('infected', axis=1)
y = df['infected']
```

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Create a Logistic Regression classifier
lr_clf = LogisticRegression()
```

```
# Train the Logistic Regression classifier
lr_clf.fit(X_train, y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression
LogisticRegression()

```
# Make predictions on the test set
y_pred = lr_clf.predict(X_test)
```

```
# Evaluate the accuracy of the Logistic Regression classifier
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8317757009345794
```

```
# Calculate the confusion matrix
confusion_mat = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(confusion_mat)
```

```
Confusion Matrix:
[[308  19]
 [ 53  48]]
```

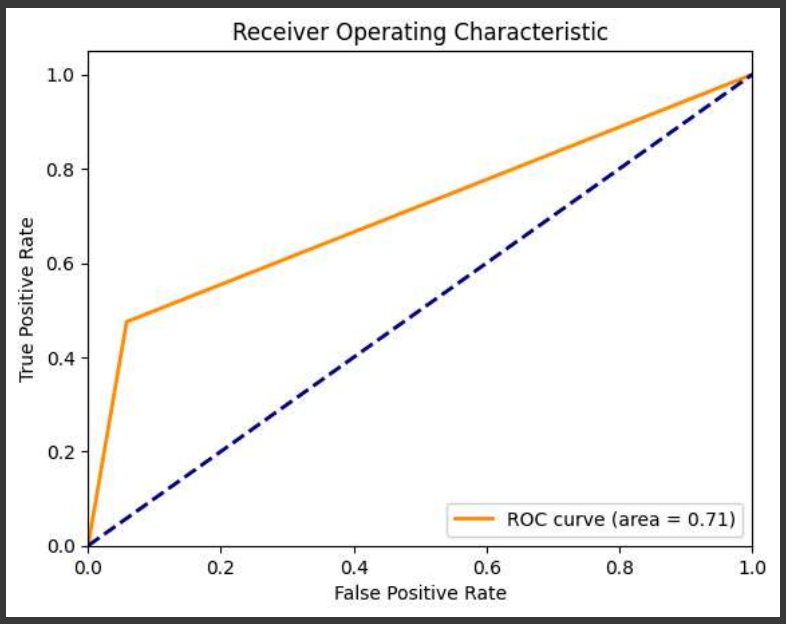
```
# Calculate the AUC (Area Under the Curve)
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
print("AUC:", roc_auc)
```

```
AUC: 0.7085717746086535
```

```
# Calculate the F1 score
f1_score_value = f1_score(y_test, y_pred)
print("F1 Score:", f1_score_value)
```

```
F1 Score: 0.5714285714285715
```

```
# Plot the ROC curve
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
         lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```




```
Start coding or generate with AI.
```