SVM Analysys

Import the data

In [2]:
```
# Import data
import pandas as pd
df = pd.read_csv("diabetes_binary_5050split_health_indicators_BRFSS2021.csv")
df.head()
```

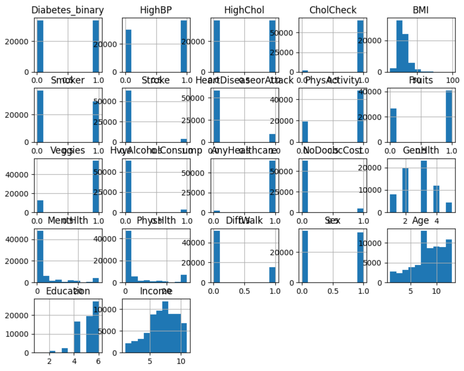
Out[2]:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
0	0.0	1	0.0	1	33.0	0.0	0.0	0.0	1	1	...	1	0.0	2.0	15.0	0.0	1.0	1	7	6.0	9.0
1	0.0	0	1.0	1	27.0	1.0	0.0	0.0	1	0	...	1	0.0	2.0	1.0	2.0	0.0	1	7	6.0	6.0
2	0.0	0	1.0	1	26.0	1.0	0.0	0.0	0	0	...	1	0.0	3.0	0.0	30.0	0.0	1	13	4.0	3.0
3	0.0	0	0.0	1	19.0	1.0	0.0	0.0	1	1	...	1	0.0	3.0	0.0	0.0	0.0	0	11	5.0	7.0
4	0.0	1	0.0	1	37.0	0.0	0.0	0.0	1	1	...	1	0.0	2.0	0.0	0.0	0.0	0	5	5.0	3.0

5 rows × 22 columns

In [3]:
```
# Import the libraries
import numpy as np
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, roc_curve, auc
import matplotlib.pyplot as plt

# Visualize the variables
df.hist(figsize=(10, 8))
plt.show()
```



In [4]:
```
X = df.drop('Diabetes_binary', axis=1)
y = df['Diabetes_binary']
```

In [5]:
```
# 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)
```

Linear Kernel

In [6]:
```
# Create an SVM model
svm_linear = SVC(C=0.01, kernel='linear') # Linear kernel
```

In [7]:
```
# Train the model
svm_linear.fit(X_train, y_train)
```

Out[7]:
```
SVC(C=0.01, kernel='linear')
```

In [8]:
```
# Set the ROC Curve results
y_test_score = svm_linear.decision_function(X_test)

fpr_l, tpr_l, _ = roc_curve(y_test, y_test_score)
roc_auc_l = auc(fpr_l, tpr_l)
```

In [14]:
```
# Performs the Classification Report
y_pred_l = svm_linear.predict(X_test)

report_l = classification_report(y_test, y_pred_l)
print(report_l)
```

```
              precision    recall  f1-score   support

         0.0       0.75      0.69      0.72      6614
         1.0       0.72      0.78      0.75      6814

    accuracy                           0.73     13428
   macro avg       0.74      0.73      0.73     13428
weighted avg       0.74      0.73      0.73     13428
```

Non-linear Kernel

In [10]:
```
# Create an SVM model
svm_rbf = SVC(C=100, gamma = 0.001, kernel='rbf')  # Non-linear kernel
```

In [11]:
```
# Train the model
svm_rbf.fit(X_train, y_train)
```

Out[11]:
```
SVC(C=100, gamma=0.001)
```

In [12]:
```
# Set the ROC Curve results
y_test_score = svm_rbf.decision_function(X_test)

fpr_n, tpr_n, _ = roc_curve(y_test, y_test_score)
roc_auc_n = auc(fpr_n, tpr_n)
```

In [15]:
```
# Performs the Classification Report
y_pred_n = svm_rbf.predict(X_test)

report_n = classification_report(y_test, y_pred_n)
print(report_n)
```

```
              precision    recall  f1-score   support

         0.0       0.78      0.67      0.72      6614
         1.0       0.72      0.81      0.76      6814

    accuracy                           0.74     13428
   macro avg       0.75      0.74      0.74     13428
weighted avg       0.75      0.74      0.74     13428
```

ROC Curves

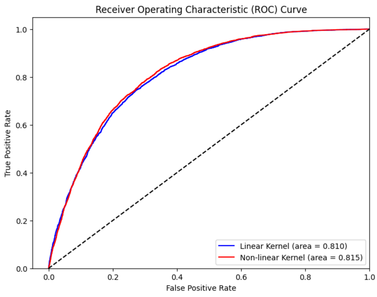
In [19]:
```
# Plot both ROC curves on the same graph
plt.figure(figsize=(8, 6))

# Linear Kernel ROC curve
plt.plot(fpr_l, tpr_l, label='Linear Kernel (area = %0.3f)' % roc_auc_l, color='b')

# Non-linear Kernel ROC curve
plt.plot(fpr_n, tpr_n, label='Non-linear Kernel (area = %0.3f)' % roc_auc_n, color='r')

# Add details to the plot
plt.plot([0, 1], [0, 1], 'k--')
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")

plt.show()
```



Grid Search - Linear Kernel

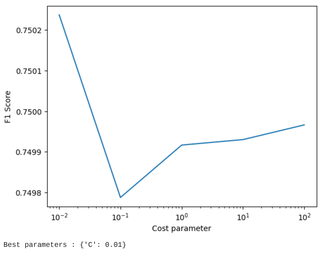
In [5]:
```
# Select the optimal C parameter by cross-validation
tuned_parameters = [{'C': [0.01, 0.1, 1, 10, 100]}]
clf = GridSearchCV(SVC(kernel='linear'), tuned_parameters, cv=2, n_jobs=-1, scoring='f1')
clf.fit(X_train, y_train)

cv_results = pd.DataFrame(clf.cv_results_)

# Plots the results
plt.plot(cv_results['param_C'], cv_results['mean_test_score'])
plt.xscale(value='log')
plt.ylabel('F1 Score')
plt.xlabel('Cost parameter')

plt.show()

# Prints the best parameters
print('Best parameters : ' + str(clf.best_params_))
```




```
Best parameters : {'C': 0.01}
```

Grid Search - Non-linear Kernel

In [6]:
```
# Select the optimal C and gamma parameters by cross-validation
tuned_parameters = [{'C': [0.01, 0.1, 1, 10, 100],
                     'gamma': [0.001, 0.01, 0.1, 1, 10]}]
clf = GridSearchCV(SVC(kernel='rbf'), tuned_parameters, cv=2, n_jobs=-1, scoring='f1')
clf.fit(X_train, y_train)
```

```
# Prints the best parameters
print('Best parameters : ' + str(clf.best_params_))
```

```
Best parameters : {'C': 100, 'gamma': 0.001}
```