Chetna Mundra - 1BM21AI036 - CYA AAT

Importing libraries

```
In [8]:
            from imblearn.over_sampling import RandomOverSampler, ADASYN
            from collections import Counter
            import matplotlib.pyplot as plt
            import seaborn as sns
          6 import pandas as pd
          7
            import numpy as np
          8
           from sklearn.model selection import train test split, KFold,
            GridSearchCV, ParameterGrid, cross_val_score
            from sklearn.linear_model import LogisticRegression
         11 | from sklearn.tree import DecisionTreeClassifier
         12 #from sklearn.ensemble import RandomForestClassifier,
            AdaBoostClassifier
         13 #from sklearn.neighbors import KNeighborsClassifier
         14 | from sklearn.preprocessing import MinMaxScaler, PowerTransformer,
            StandardScaler
         15 | from sklearn.metrics import roc_auc_score
         16 from sklearn.svm import SVC
         17
            from sklearn import metrics
         18
         19
         20
         21
         22
            import warnings
            warnings.filterwarnings("ignore")
```

Reading Data

dtype: int64

Train test split

```
In [14]: 1 # Train test split
2 y = df.pop("Class")
3 X = df
4
5 # Using stratify=y for splitting data into stratified fashion
6 X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.75, stratify=y, random_state=42)
```

Results

Model Evaluation Function

```
In [21]:
           1
             def model_evaluation(y_pred_proba_test):
                    Report for different thresholds
           2
           3
                  thresholds = [i * 0.1 \text{ for } i \text{ in } range(0, 10)]
                  print("-----Results------
           4
              ----")
           5
             #
                   best ROC score initialisation
           6
                  best_roc_score = 0
           7
           8
                    Iterating through every threshold from 0.1 to 0.9
           9
                  for threshold in thresholds:
          10
                      y_pred = np.where(y_pred_proba_test[:, 1] > threshold, 1, 0)
          11
                  Calculating different metrics
          12
                      accuracy = str(round(metrics.accuracy_score(y_test, y_pred),
             3))
                      precision = str(round(metrics.precision_score(y_test,
          13
             y_pred), 3))
                      recall = str(round(metrics.recall_score(y_test, y_pred), 3))
          14
                      roc_auc = str(round(metrics.roc_auc_score(y_test, y_pred),
          15
             3))
                 Setting the best roc score, threshold, recall scores.
          16
          17
                      if float(roc_auc) > best_roc_score:
                          best_roc_score = float(roc_auc)
          18
          19
                          best threshold = threshold
                          best_recall_score = recall
          20
          21
                 printing the results for every threshold
                      print("-----for Test with threshold", round(threshold,
          22
             2), "----")
                      print("accuracy\tprecision\trecall\t\troc_auc")
          23
                      print("\t\t".join([accuracy, precision, recall, roc_auc]))
          24
                      print("\n")
          25
          26
                 Confusion Matrix
                      print("\t\tCONFUSION MATRIX")
          27
          28
                      confusion_matrix =
              pd.DataFrame(metrics.confusion_matrix(y_test, y_pred),
          29
                                                      columns=['Predicted
             Negative', 'Predicted Positive'],
          30
                                                      index=['Actual Negative',
              'Actual Positive'])
                      print(confusion_matrix)
          31
          32
                      print("\n")
                  print("BEST ROC AUC SCORE is ", best roc score, "at the
          33
             threshold", best_threshold)
          34
                  return best roc score, best threshold, best recall score
```

Draw ROC

```
In [22]:
           1 def draw_roc( actual, probs ):
                 fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
           2
                                                            drop_intermediate =
             False )
           4
                  auc_score = metrics.roc_auc_score( actual, probs )
           5
                  plt.figure(figsize=(5, 5))
           6
                 plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score
           7
                 plt.plot([0, 1], [0, 1], 'k--')
           8
                 plt.xlim([0.0, 1.0])
                 plt.ylim([0.0, 1.0])
           9
                 plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
          10
          11
                 plt.ylabel('True Positive Rate')
          12
                 plt.title('Receiver operating characteristic')
          13
                 plt.legend(loc="lower right")
          14
                  plt.show()
```

XG BOOST

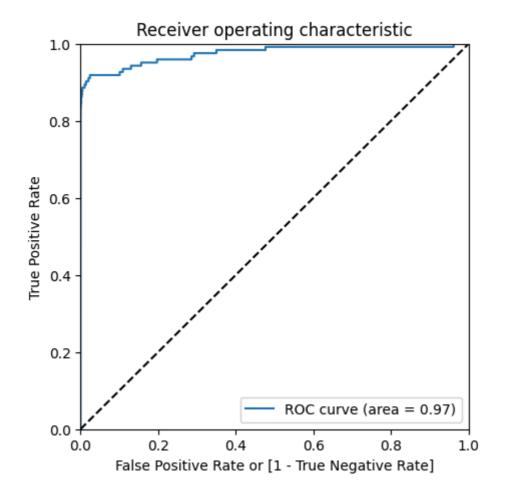
```
In [24]:
           1 # Import necessary libraries
           2 import xgboost as xgb
           3 from sklearn.model_selection import GridSearchCV, KFold
           4 from sklearn.pipeline import make pipeline
             import pandas as pd
           5
           6
           7 # Initialize the XGBoost model
           8 def initialize_xgboost():
                  return xgb.XGBClassifier(eval_metric='logloss', n_estimators=100,
             max depth=3)
          10
          11 # Train the model using GridSearchCV with pipeline
          12 def train_xgboost(X_train, y_train, cv):
          13
                  xgb_model = initialize_xgboost()
                  pipeline = make_pipeline(xgb_model)
          14
          15
                  param_grid = {
                      'xgbclassifier__n_estimators': [50, 100],
          16
                      'xgbclassifier__max_depth': [3, 5]
          17
          18
          19
                  grid = GridSearchCV(pipeline, param_grid=param_grid,
              scoring='roc_auc', cv=cv, n_jobs=-1, verbose=1000)
          20
                  grid.fit(X_train, y_train)
                  print("BEST GRID SCORE", grid.best_score_)
          21
                  print("BEST GRID PARAMS")
          22
          23
                  print(grid.best_params_)
          24
                  return grid.best_estimator_.named_steps['xgbclassifier']
          25
          26 # Evaluate the model
          27 def evaluate_xgboost(model, X_test, y_test):
          28
                  # Get the predicted probabilities for both classes
          29
                  y_pred_proba_test = model.predict_proba(X_test)
                  # Pass the full probability array to model_evaluation
          30
                  best_roc_score, best_threshold, best_recall_score =
              model_evaluation(y_pred_proba_test)
                  return best_roc_score, best_threshold, best_recall_score
          32
          33
          34 # Example usage
          35 \text{ cv} = \text{KFold}(3)
          36 best_xgb_model = train_xgboost(X_train, y_train, cv)
             best_roc_score, best_threshold, best_recall_score =
              evaluate xgboost(best xgb model, X test, y test)
          38
          39 # Update results dataframe
          40 | data = pd.DataFrame([['XGBOOST', best_threshold, best_recall_score,
              best_roc_score]], columns=results.columns)
          41 results = pd.concat([results, data], ignore_index=False)
          42
          43 # Optional: Plot ROC curve
             draw_roc(y_test, best_xgb_model.predict_proba(X_test)[:, 1])
```

{	max_depth': 3 Re	3, 'xgbclassifi esults	ern_estimators':	50 -
for Te	est with thresh	nold 0.0		
accuracy 0.002	precision	recall	roc_auc	
0.002	0.002	1.0	0.5	
	CONFUSION MATE)TV		
		kia gative Predict	ed Positive	
Actual Negative	Treateted Neg	0	71079	
Actual Positive		0	123	
_				
for Te				
accuracy 0.999				
0.555	0.73	0.029	0.914	
	CONFUSION MATE	RIX		
		gative Predict	ed Positive	
Actual Negative			34	
Actual Positive		21	102	
for Te	est with thresh	nold 0.2		
accuracy				
0.999				
	CONFUSION MATE		ad Dagitina	
Actual Negative		gative Predict 71062	.eu Positive 17	
Actual Positive		25	98	
necual rosicive		23	30	
for Te				
accuracy	precision	recall	roc_auc	
0.999	0.882	0.789	0.894	
	CONFUSION MATE	RIX		
		gative Predict	ed Positive	
Actual Negative	•	71066	13	
Actual Positive		26	97	
for Te	oct with throck	old 0 4		
accuracy				
1.0	•	0.78	_	
	CONFLICTOR MATE	DTV		
	CONFUSION MATE		and Dositive	
Actual Negative	rredicted Neg	gative Predict 71071	ed Positive 8	
ACTUAL NEEDTIVE		\ T \ \ T	ŏ	
Actual Positive		27	96	

-----for Test with threshold 0.5 -----

accuracy 1.0	precision 0.931	reca 0.77		roc_auc 0.886
Actual Negative Actual Positive			Predicted	Positive 7 95
for T accuracy 1.0		reca		roc_auc
	CONFUSION MA	TRTX		
Actual Negative Actual Positive	Predicted No		Predicted	Positive 5 93
for T accuracy 0.999		reca	11	roc_auc
Actual Negative Actual Positive			Predicted	Positive 3 90
for T accuracy 0.999	precision	reca	11	roc_auc
Actual Negative Actual Positive			Predicted	Positive 3 85
for T accuracy 0.999		shold 0.9 reca 0.59	11	roc_auc 0.797
Actual Negative Actual Positive			Predicted	Positive 1 73

BEST ROC AUC SCORE is 0.914 at the threshold 0.1



Random Forest

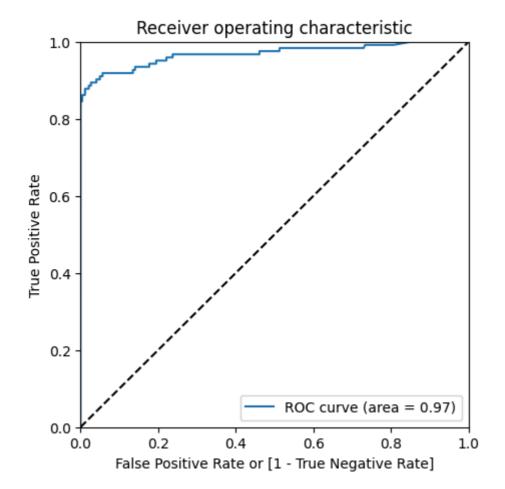
```
In [28]:
           1 # Import necessary libraries
           2 from sklearn.ensemble import RandomForestClassifier
           3 from sklearn.model_selection import train_test_split
           4 import pandas as pd
           6 # Train a simple Random Forest model
          7 def train_random_forest(X_train, y_train):
                 rf_model = RandomForestClassifier(n_estimators=50, max_depth=10,
             random_state=42)
          9
                 rf model.fit(X train, y train)
          10
                 return rf_model
          11
          12 # Example usage
          13 X_train, X_test, y_train, y_test = train_test_split(X, y,
             test_size=0.25, stratify=y, random_state=42)
          14 best_rf_model = train_random_forest(X_train, y_train)
          15
          16 # Evaluate the model using the existing function
             y_pred_proba_test = best_rf_model.predict_proba(X_test)
          17
          18 best_roc_score, best_threshold, best_recall_score =
             model_evaluation(y_pred_proba_test)
          19
          20 # Update results dataframe
          21 data = pd.DataFrame([['RANDOM FOREST', best_threshold,
             best_recall_score, best_roc_score]], columns=results.columns)
          22 results = pd.concat([results, data], ignore_index=False)
          23
          24 # Optional: Plot ROC curve
          25 draw_roc(y_test, y_pred_proba_test[:, 1])
          26
```

Results						
	est with thresho					
	precision	recall	roc_auc			
0.002	0.002	1.0	0.5			
	CONFUSION MATRIX	X				
		tive Predicted	Positive			
Actual Negative		1	71078			
Actual Positive		0	123			
7.00001 7.0010170		· ·	113			
for T	est with thresho	ld 0 1	_			
accuracy	precision	U 01C	a one			
0.999	0.707	0.846	0.922			
	CONFUSION MATRIX					
	_	tive Predicted				
Actual Negative	7:	1036	43			
Actual Positive		19	104			
for To	est with thresho	ld 0.2	_			
	precision					
	0.811					
0.333	0.011	0.037	0.515			
	CONFUSION MATRIX	v				
			D141			
		tive Predicted				
Actual Negative	/:	1055	24			
Actual Positive		20	103			
for T	est with thresho	ld 0.3	-			
accuracy	precision 0.877	recall	roc_auc			
0.999	0.877	0.813	0.906			
	CONFUSION MATRIX	Κ				
		tive Predicted	Positive			
Actual Negative	•	1065	14			
Actual Positive	,.	23	100			
ACCUAL POSICIVE		23	100			
C T		1 1 0 4				
	est with thresho					
	precision		roc_auc			
1.0	0.933	0.789	0.894			
	CONFUSION MATRIX	X				
	Predicted Negat	tive Predicted	Positive			
Actual Negative		1072	7			
Actual Positive		26	97			
for To	est with thresho	ld 0.5	_			
			roc_auc			
1.0	0.94	0.764	0.882			
T.0	U. 34	0.704	0.002			

CONFUSION MATRIX

			AIKIX		
		Predicted	Negative	Predicted	Positive
Actual	Negative		71073		6
Actual	Positive		29		94
	for Te	est with thr	eshold 0.0	б	
		precision			
0.999	· y	0.966	0 60	Ω	0 95
0.555		0.500	0.05	,	0.05
		CONFLICTON	A T D T \		
		CONFUSION M			
		Predicted		Predicted	Positive
Actual	Negative		71076		3
Actual	Positive		37		86
	for Te	est with thr	eshold 0.	7	
		precision			
0.999	· y			1 1	
0.999		0.976	0.05		0.025
		CONFUSION M			
		Predicted	Negative	Predicted	Positive
Actual	Negative		71077		2
			. =		
Actual	Positive		43		80
Actual	Positive				80
Actual	Positive				80
		est with thr	43	8	
	for Te	est with thr	43 eshold 0.8		
accurac	for Te	precision	43 eshold 0.8 reca	11	 roc_auc
	for Te		43 eshold 0.8 reca	11	
accurac	for Te	precision	43 eshold 0.8 reca	11	 roc_auc
accurac	for Te	precision 0.971	43 eshold 0.8 reca 0.53	11	 roc_auc
accurac	for Te	precision 0.971 CONFUSION M	43 eshold 0.8 reca 0.53	11 7	 roc_auc 0.768
accurac	for Te	precision 0.971 CONFUSION M	43 eshold 0.8 reca 0.53	11	 roc_auc 0.768
accurac 0.999	for Te	precision 0.971 CONFUSION M Predicted	43 eshold 0.3 reca 0.53 ATRIX Negative	ll 7 Predicted	roc_auc 0.768 Positive
accurac 0.999	for Te	precision 0.971 CONFUSION M	43 eshold 0.3 reca 0.53 ATRIX Negative	ll 7 Predicted	roc_auc 0.768 Positive 2
accurac 0.999	for Te	precision 0.971 CONFUSION M Predicted	43 reshold 0.3 reca 0.53 ATRIX Negative 71077	ll 7 Predicted	roc_auc 0.768 Positive
accurac 0.999	for Te	precision 0.971 CONFUSION M Predicted	43 reshold 0.3 reca 0.53 ATRIX Negative 71077	ll 7 Predicted	roc_auc 0.768 Positive 2
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted	43 reshold 0.3 reca 0.53 ATRIX Negative 71077 57	ll 7 Predicted	roc_auc 0.768 Positive 2 66
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted	eshold 0.3 reca 0.53 ATRIX Negative 71077 57	11 7 Predicted 9	roc_auc 0.768 Positive 2 66
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision	eshold 0.3 reca 0.53 ATRIX Negative 71077 57 eshold 0.9	11 7 Predicted 9	roc_auc 0.768 Positive 2 66 roc_auc
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted	eshold 0.3 reca 0.53 ATRIX Negative 71077 57	11 7 Predicted 9	roc_auc 0.768 Positive 2 66
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision	eshold 0.3 reca 0.53 ATRIX Negative 71077 57 eshold 0.9	11 7 Predicted 9	roc_auc 0.768 Positive 2 66 roc_auc
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision	eshold 0.3 reca 0.53 ATRIX Negative 71077 57 eshold 0.9	11 7 Predicted 9	roc_auc 0.768 Positive 2 66 roc_auc
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision	eshold 0.3 reca 0.53 ATRIX Negative 71077 57 eshold 0.9 reca 0.40	11 7 Predicted 9	roc_auc 0.768 Positive 2 66 roc_auc
accurac 0.999 Actual Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision 0.98 CONFUSION M	43 eshold 0.3 recal 0.53 ATRIX Negative 71077 57 eshold 0.9 recal 0.40	11 7 Predicted 9 11 7	roc_auc 0.768 Positive 2 66 roc_auc 0.703
accurac 0.999 Actual Actual	for Te	precision 0.971 CONFUSION M Predicted est with thr precision 0.98 CONFUSION M	43 eshold 0.3 reca 0.53 ATRIX Negative 71077 57 eshold 0.9 reca 0.40	11 7 Predicted 9	roc_auc 0.768 Positive 2 66 roc_auc 0.703
Actual Actual accurac 0.999 Actual	for Te y Negative Positive	precision 0.971 CONFUSION M Predicted est with thr precision 0.98 CONFUSION M	43 eshold 0.3 recal 0.53 ATRIX Negative 71077 57 eshold 0.9 recal 0.40	11 7 Predicted 9 11 7	roc_auc 0.768 Positive 2 66 roc_auc 0.703

BEST ROC AUC SCORE is 0.922 at the threshold 0.1



Using SMOTE analsyis

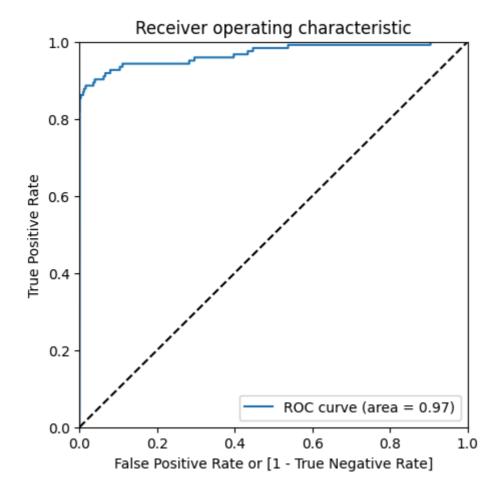
```
In [29]:
           1 # Import necessary libraries
           2 from sklearn.linear_model import LogisticRegression
           3 from sklearn.model_selection import train_test_split
           4 from imblearn.over_sampling import SMOTE
           5
             import pandas as pd
          7 # Apply SMOTE to balance the dataset
          8 def apply_smote(X_train, y_train):
          9
                 smote = SMOTE(random_state=42)
                 X resampled, y resampled = smote.fit resample(X train, y train)
          10
          11
                 return X_resampled, y_resampled
          12
          13 # Train a simple Logistic Regression model
          14 | def train_logistic_regression(X_train, y_train):
                 lr_model = LogisticRegression(max_iter=1000, random_state=42)
          15
          16
                 lr_model.fit(X_train, y_train)
          17
                 return lr_model
          18
          19 # Example usage
          20 | X_train, X_test, y_train, y_test = train_test_split(X, y,
             test_size=0.25, stratify=y, random_state=42)
          21 X_resampled, y_resampled = apply_smote(X_train, y_train)
          22
          23 best_lr_model = train_logistic_regression(X_resampled, y_resampled)
          24
          25 # Evaluate the model using the existing function
          26  y_pred_proba_test = best_lr_model.predict_proba(X_test)
          27 best_roc_score, best_threshold, best_recall_score =
             model_evaluation(y_pred_proba_test)
          28
          29 # Update results dataframe
          30 data = pd.DataFrame([['LOGISTIC REGRESSION (SMOTE)', best_threshold,
             best_recall_score, best_roc_score]], columns=results.columns)
          31 | results = pd.concat([results, data], ignore_index=False)
          32
          33 # Optional: Plot ROC curve
          34 draw_roc(y_test, y_pred_proba_test[:, 1])
          35
```

	Res	ults	
		old 0.0	
accuracy	precision	recall	roc_auc
	0.002		0.5
	CONFUSION MATRI	X	
	Predicted Nega	tive Predicted	Positive
Actual Negative		0	71079
Actual Positive		0	123
		old 0.1	
		recall	
0.879	0.013	0.943	0.911
	CONFUSION MATRI	X X	
	Predicted Nega	tive Predicted	Positive
Actual Negative	. 6	52471	8608
Actual Positive		7	116
for T	est with thresho	old 0.2	_
accuracy	precision	recall	roc auc
0.936	0.024	0.911	0.923
	CONFUSION MATRI	XX	
		tive Predicted	Positive
Actual Negative		6546	4533
Actual Positive		11	112
, le cadi i obieti			
for T	est with thresho	old 0.3	_
	precision	recall	roc_auc
	0.038	0.894	0.928
0.502	0.030	0.051	0.720
	CONFUSION MATRI	ΣX	
		tive Predicted	Positive
Actual Negative		8305	2774
Actual Positive		13	110
ACCUAL TOSICIVE		15	110
for T	est with thresho	old 0.4	_
		recall	
-	0.056	0.886	0.93
0.574	0.030	0.000	0.33
	CONFUSION MATRI	·X	
		tive Predicted	Positive
Actual Negative	_	59258	1821
Actual Negative Actual Positive		14	109
ACTUAL FUSICIVE		7-4	103
	est with thresho	old 0.5	_
	precision		
accuracy 0.982	0.078	0.886	roc_auc 0.934
U.70Z	0.0/0	0.000	U.734

Actual Negative Actual Positive		Negative	Predicted	Positive 1283 109
		14		
for T accuracy 0.987	precision	reca	11	roc_auc
	CONFUSION M	ATRIX		
Actual Negative Actual Positive	Predicted	Negative 70194 15	Predicted	Positive 885 108
for T				
accuracy 0.992	precision	reca a se	11	roc_auc
0.932	0.130	0.80	2	0.327
	CONFUSION M			
Actual Negative	Predicted	Negative 70506	Predicted	Positive 573
Actual Negative Actual Positive		17		106
for T				
accuracy 0.994				
	CONFUSION M			
	Predicted	•	Predicted	
Actual Negative Actual Positive		70672 17		407 106
Actual Positive		17		100
for T	est with thr	eshold 0.	9	
accuracy	precision	reca		roc_auc
0.996	0.268	0.86	2	0.929
	CONFUSION M	ATRIX		
Actual Negative	Predicted	Negative	Predicted	Positive

BEST ROC AUC SCORE is 0.934 at the threshold 0.5

Actual Negative Actual Positive



In [30]: 1 results.sort_values("recall", ascending=False, inplace=True)
2 # results.to_csv("results.csv", index = False) #for saving the data
 frame to csv file
3 results

\sim			$\Gamma \sim$	\sim	
()	117	г I	ι - ۷	и	
v	u	u		U	

	model_name	threshold	recall	roc_auc_score
0	LOGISTIC REGRESSION (SMOTE)	0.5	0.886	0.934000
0	RANDOM FOREST	0.1	0.846	0.922000
0	XGBOOST	0.1	0.829	0.914000
0	RANDOM FOREST	NaN	None	0.968519

Type *Markdown* and LaTeX: α^2