Hepatitis C Virus Classification

Dataset: https://archive.ics.uci.edu/dataset/571/hcv+data

Libraries import

In [ ]:

```
from utils import *
import matplotlib.pyplot as plt
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.calibration import cross_val_predict
from sklearn.neighbors import KNeighborsClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.calibration import cross_val_predict
import seaborn as sn
from sklearn.metrics import roc_curve, auc

def myPlotROCcurve(target_test, prediction, text=""):
    fpr, tpr, _ = roc_curve(target_test, prediction)
    roc_auc = auc(fpr, tpr)
    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) - ' + text)
    plt.legend(loc='lower right')
    plt.show()

def myPlotConfusionMatrix(target_test, prediction, text=""):
    conf_matrix = confusion_matrix(target_test, prediction)
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False)
    plt.xlabel('Predicted Class')
    plt.ylabel('True Class')
    plt.title(text)
    plt.show()


def myCrossValidation(classifier, cv_techique, features, target, text=""):
    accuracy = cross_val_score(
    classifier, # Classifier
    features, # Feature matrix
    target, # Target vector
    cv=cv_techique, # Cross-validation technique
    scoring="accuracy", # Loss function
    n_jobs=-1) # Use all CPU scores

    accuracy = np.mean(accuracy)
    precision = cross_val_score(
        classifier, # Classifier
        features, # Feature matrix
        target, # Target vector
        cv=cv_techique, # Cross-validation technique
        scoring="precision", # Loss function
        n_jobs=-1) # Use all CPU scores

    precision = np.mean(precision)

    recall = cross_val_score(
        classifier, # Classifier
        features, # Feature matrix
        target, # Target vector
        cv=cv_techique, # Cross-validation technique
        scoring="recall", # Loss function
        n_jobs=-1) # Use all CPU scores

    recall = np.mean(recall)

    f1 = cross_val_score(
        classifier, # Classifier
        features, # Feature matrix
        target, # Target vector
        cv=cv_techique, # Cross-validation technique
        scoring="f1", # Loss function
        n_jobs=-1) # Use all CPU scores

    f1 = np.mean(f1)

    predictions = cross_val_predict(
        classifier,
        features,
        target,
        cv=cv_techique,
        method='predict')

    return pd.DataFrame({"Accuracy": accuracy,
                         "Precision": precision,
                         "Recall": recall,
                         "F1": f1,
                         "Model": text},
                         index=[0]), predictions

def myResultFormalizer(report, text=""):

    return pd.DataFrame({"Accuracy": float(report['accuracy']),
                         "Precision": float(report['0']['precision']),
                         "Recall": float(report['0']['recall']),
                         "F1": float(report['0']['f1-score']),
                         "Model": text},
                         index=[0])
```

Data import and manipulation

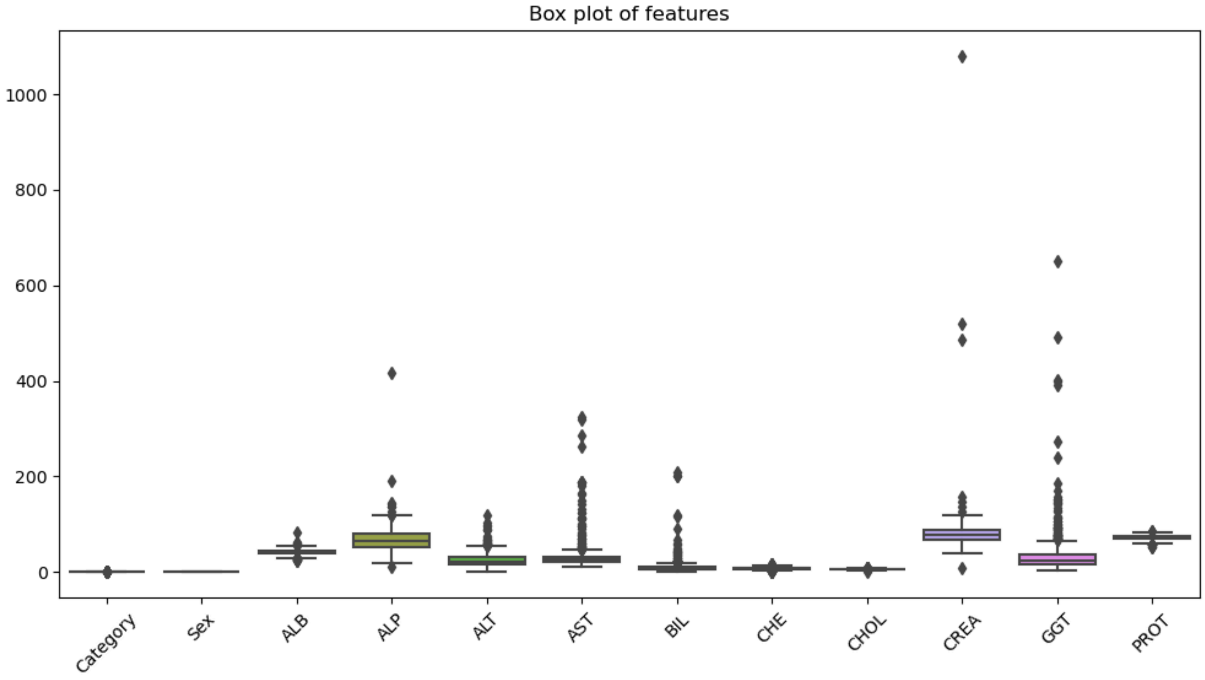
In [ ]:

```
dataframe = pd.read_csv('hcvdat0.csv')

dataframe.drop("Unnamed: 0", axis=1, inplace=True)

dataframe.dropna(inplace=True)


scale_mapper = {
    "0=Blood Donor": 0,
    "0s=suspect Blood Donor": 2,
    "1=Hepatitis": 1,
    "2=Fibrosis": 1,
    "3=Cirrhosis": 1,}

# Deviding blood types in two types
dataframe['Category'] = dataframe['Category'].replace(scale_mapper)

dataframe['Sex'] = dataframe['Sex'].replace({"m":0, "f":1})

# Removal of suspect blood donor becaouse they are not useful for model
dataframe = dataframe[dataframe['Category'] != 2]

# Removal of wrong categorized data
dataframe = dataframe[dataframe['Category'].isin([0, 1])]
dataframe = dataframe[dataframe['Sex'].isin([0, 1])]

# Define the age ranges and labels for each category
age_bins = [0, 18, 30, 40, 50, 60, 70, 120]  # Define the age bins
age_labels = [0, 1, 2, 3, 4, 5, 6]  # Define the labels for each age group

# Categorize ages into age groups
dataframe['Age'] = pd.cut(dataframe['Age'], bins=age_bins, labels=age_labels)

dataframe.head(20)

dataframe.to_csv("wrangled_data.csv")
```

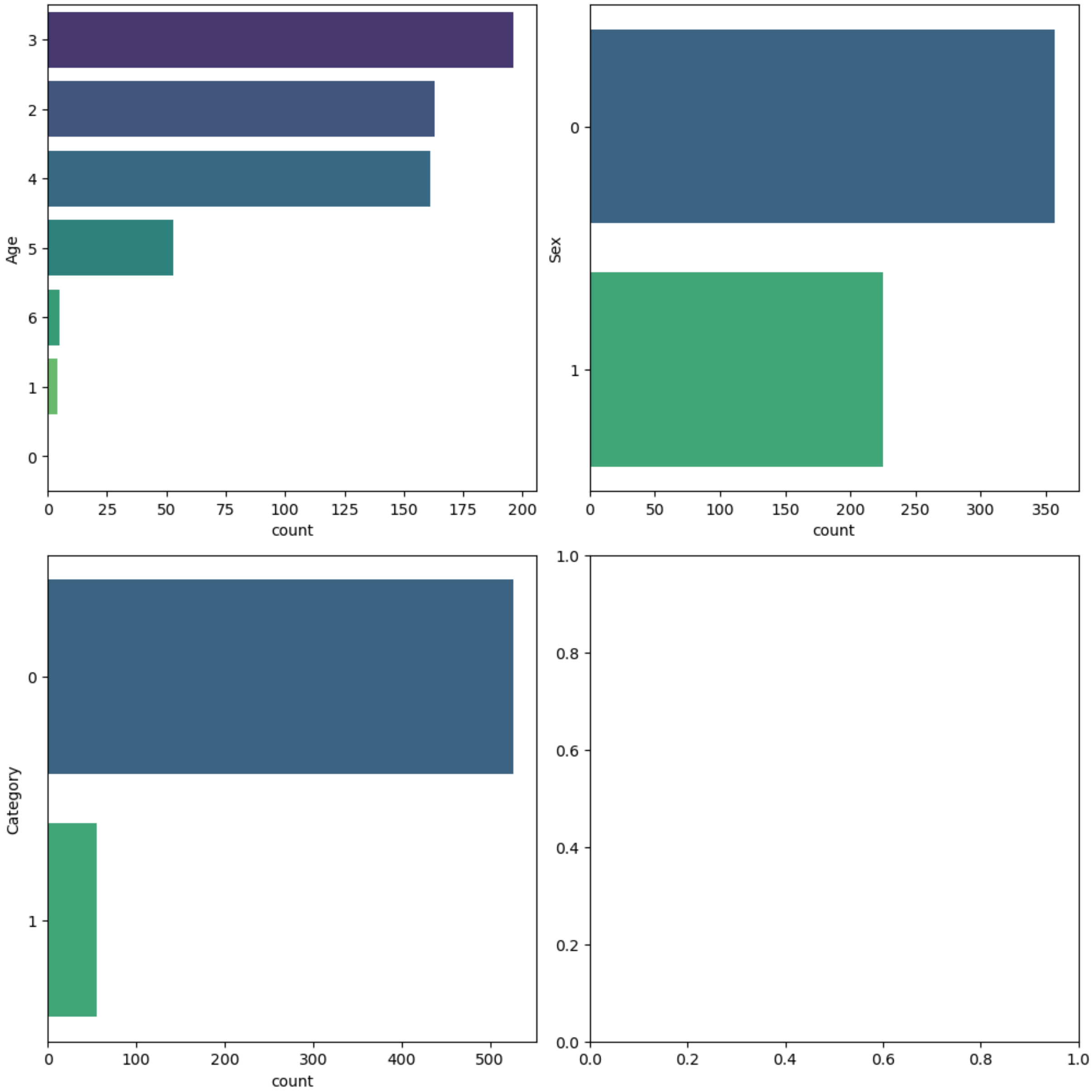
Outlier detection

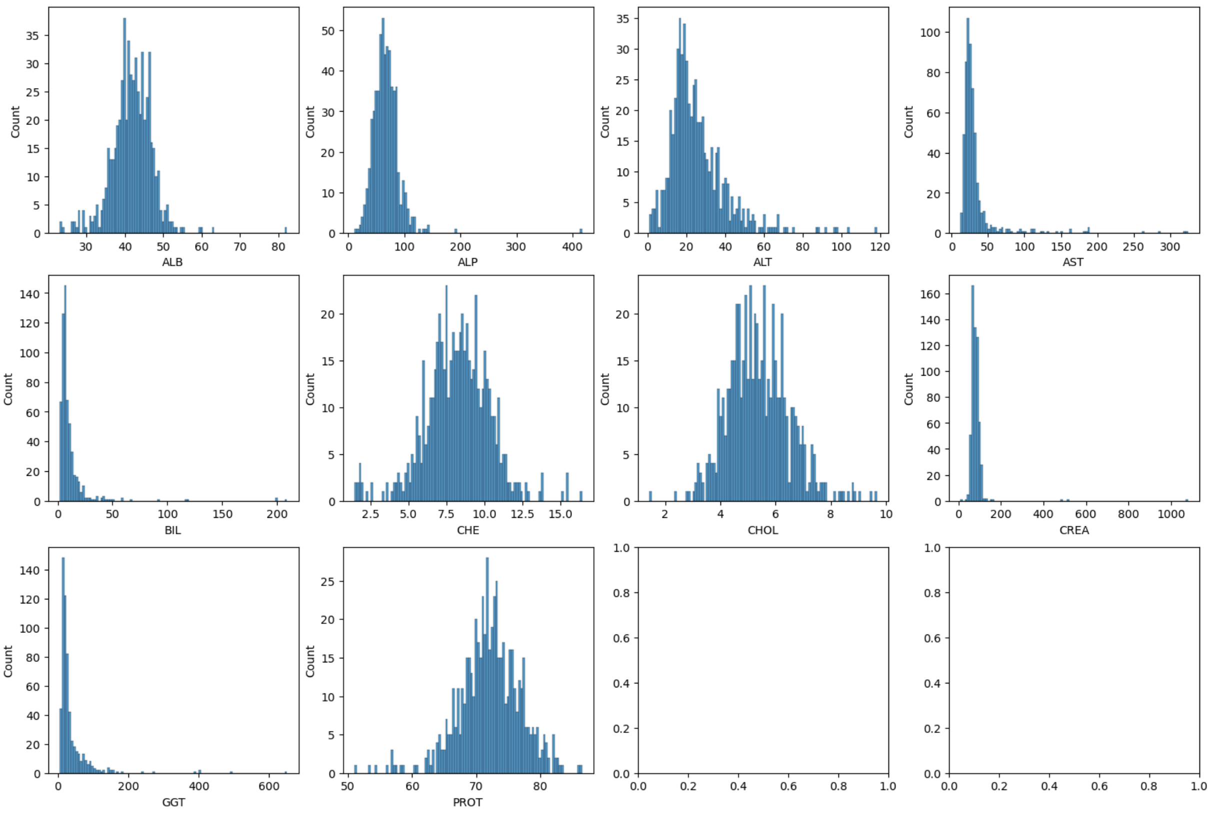
In [ ]:

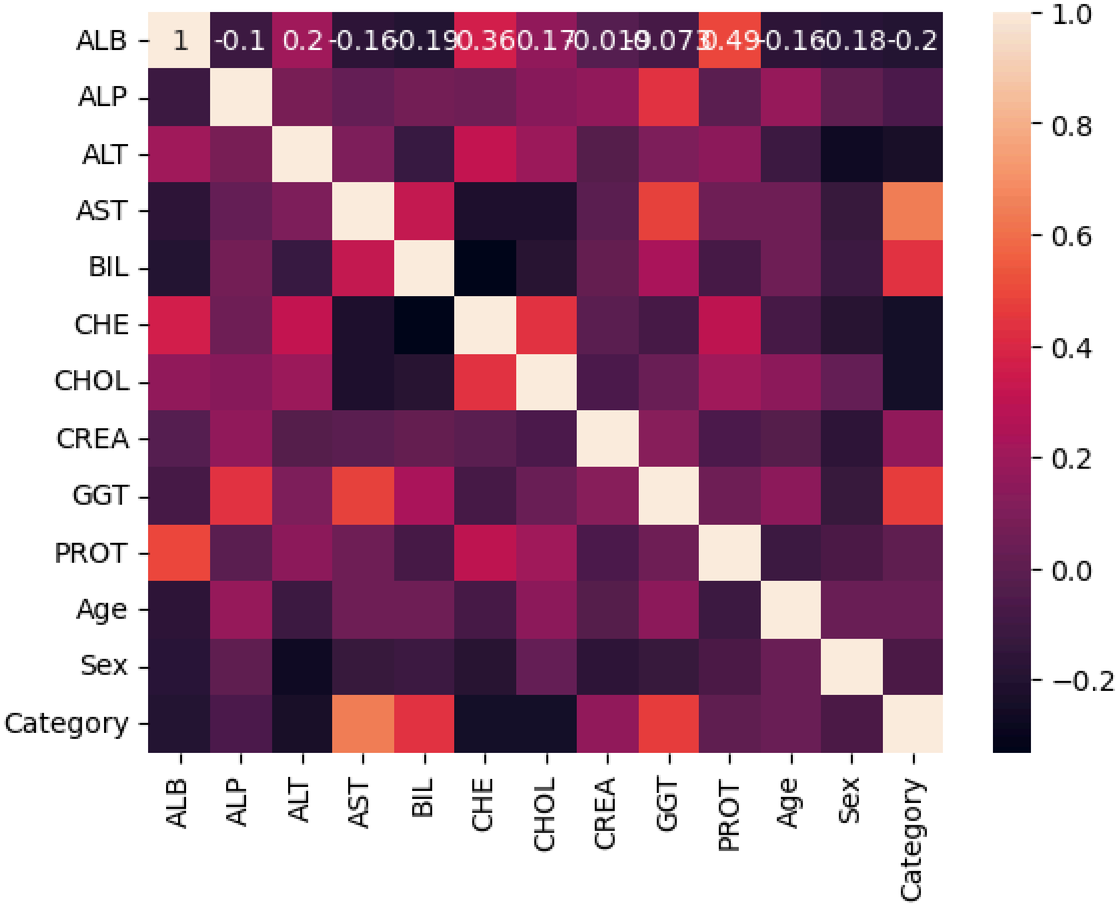
```
# Create a box plot for each feature
plt.figure(figsize=(12, 6))
sns.boxplot(data=dataframe)
plt.title('Box plot of features')
plt.xticks(rotation=45)
plt.show()

fig, axs = plt.subplots(2, 2, figsize=(10,10), constrained_layout=True)
categorical = ['Age', 'Sex', 'Category']
for i, f in enumerate(categorical):
    sns.countplot(y=f, data=dataframe, ax=axs[i//2][i%2], order=dataframe[f].value_

fig, axs = plt.subplots(3, 4, figsize=(15, 10), constrained_layout=True)
numerical = ['ALB', 'ALP', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT', 'PROT
for i, f in enumerate(numerical):
    sns.histplot(x=f, data=dataframe, ax=axs[i//4][i%4], bins=100)  # Adjusted inde
plt.show()

corr_matrix = dataframe[[*numerical, *categorical]].corr()
sn.heatmap(corr_matrix, annot=True)
plt.show()
print(corr_matrix)
```



```
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\categorical.py:641:
FutureWarning: The default of observed=False is deprecated and will be changed to Tr
ue in a future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Age
3
2
4
5
6
1
0
0
25
50
75
100
125
150
175
200
count
Sex
0
1
0
50
100
150
200
250
300
350
count
Category
0
1
0
100
200
300
400
500
count
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


Count
ALB
30 40 50 60 70 80
0 5 10 15 20 25 30 35
ALP
0 100 200 300 400
0 10 20 30 40 50
ALT
0 20 40 60 80 100 120
0 5 10 15 20 25 30 35
AST
0 50 100 150 200 250 300
0 20 40 60 80 100
BIL
0 50 100 150 200
0 20 40 60 80 100 120 140
CHE
2.5 5.0 7.5 10.0 12.5 15.0
0 5 10 15 20
CHOL
2 4 6 8 10
0 5 10 15 20
CREA
0 200 400 600 800 1000
0 20 40 60 80 100 120 140 160
GGT
0 200 400 600
0 20 40 60 80 100 120 140
PROT
60 70 80
0 5 10 15 20 25
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.2 0.4 0.6 0.8 1.0

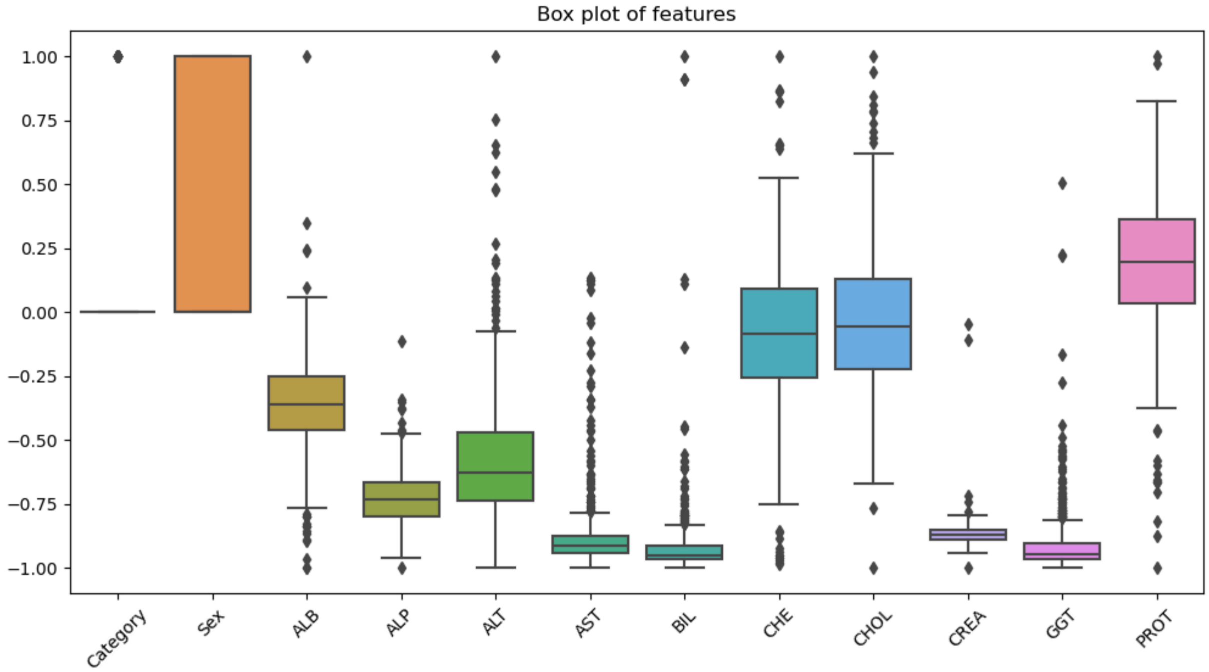

ALB
ALP
ALT
AST
BIL
CHE
CHOL
CREA
GGT
PROT
Age
Sex
Category
1 -0.1 0.2 -0.16 -0.19 0.36 0.17 -0.019 -0.073 0.49 -0.16 -0.18 -0.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2

```
               ALB       ALP       ALT       AST       BIL       CHE  \
ALB       1.000000 -0.103431  0.200235 -0.161389 -0.194822  0.364617
ALP      -0.103431  1.000000  0.078155  0.027222  0.066974  0.043800
ALT       0.200235  0.078155  1.000000  0.100191 -0.126918  0.311920
AST      -0.161389  0.027222  0.100191  1.000000  0.321066 -0.224441
BIL      -0.194822  0.066974 -0.126918  0.321066  1.000000 -0.330494
CHE       0.364617  0.043800  0.311920 -0.224441 -0.330494  1.000000
CHOL      0.167053  0.135502  0.184870 -0.211018 -0.187060  0.428312
CREA     -0.018850  0.167393 -0.024710 -0.009114  0.019625 -0.013163
GGT      -0.073379  0.428599  0.102898  0.481591  0.234191 -0.074410
PROT      0.493732 -0.014861  0.146857  0.048419 -0.071996  0.297580
Age      -0.159642  0.173311 -0.114250  0.057835  0.048684 -0.085646
Sex      -0.178748  0.003135 -0.272588 -0.134649 -0.111291 -0.185438
Category -0.203710 -0.062378 -0.233696  0.645313  0.442584 -0.248236

              CHOL      CREA       GGT      PROT       Age       Sex  Category
ALB       0.167053 -0.018850 -0.073379  0.493732 -0.159642 -0.178748 -0.203710
ALP       0.135502  0.167393  0.428599 -0.014861  0.173311  0.003135 -0.062378
ALT       0.184870 -0.024710  0.102898  0.146857 -0.114250 -0.272588 -0.233696
AST      -0.211018 -0.009114  0.481591  0.048419  0.057835 -0.134649  0.645313
BIL      -0.187060  0.019625  0.234191 -0.071996  0.048684 -0.111291  0.442584
CHE       0.428312 -0.013163 -0.074410  0.297580 -0.085646 -0.185438 -0.248236
CHOL      1.000000 -0.060087  0.031976  0.201525  0.137555  0.025227 -0.252205
CREA     -0.060087  1.000000  0.128460 -0.061710 -0.032940 -0.160133  0.166441
GGT       0.031976  0.128460  1.000000  0.049717  0.138731 -0.133407  0.461679
PROT      0.201525 -0.061710  0.049717  1.000000 -0.108158 -0.069882  0.006535
Age       0.137555 -0.032940  0.138731 -0.108158  1.000000  0.032236  0.037018
Sex       0.025227 -0.160133 -0.133407 -0.069882  0.032236  1.000000 -0.067596
Category -0.252205  0.166441  0.461679  0.006535  0.037018 -0.067596  1.000000
```

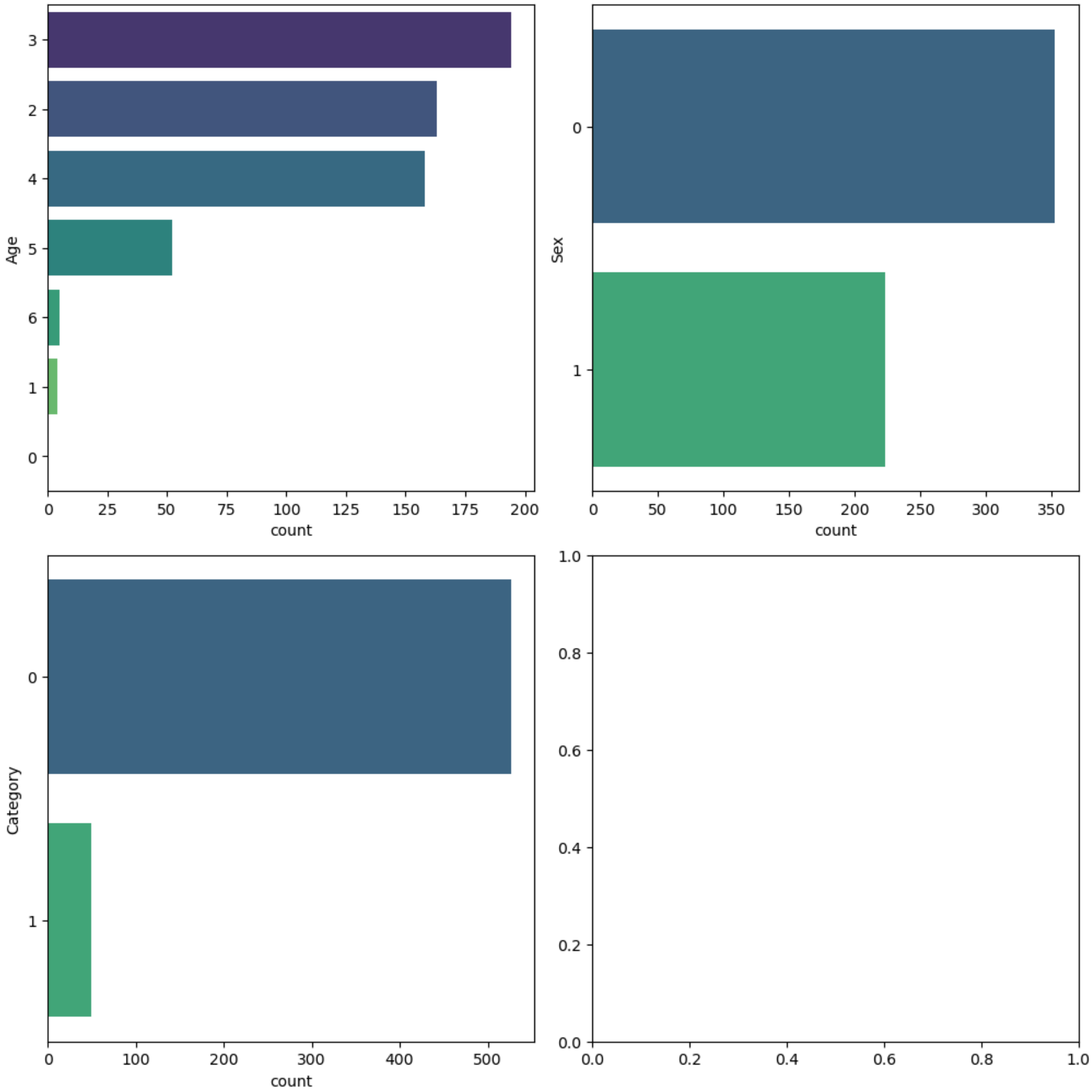
In [ ]:

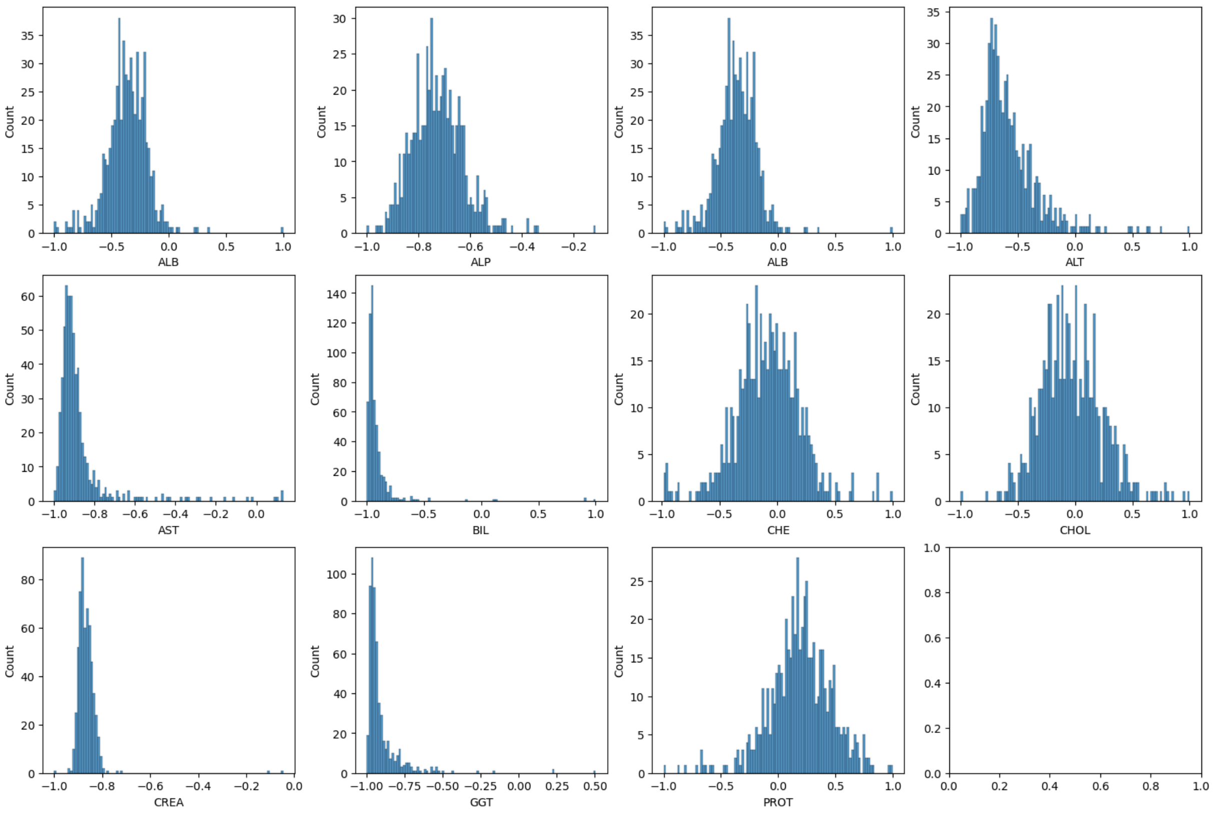
```
columns_to_scale = ['ALB', 'ALP', 'ALB', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA'

scaler = MinMaxScaler(feature_range=(-1, 1))
dataframe[columns_to_scale] = scaler.fit_transform(dataframe[columns_to_scale])

outlier_detector = EllipticEnvelope(contamination=.009)

# Fit detector
outlier_detector.fit(dataframe[columns_to_scale])

# Predict outliers
outliers = outlier_detector.predict(dataframe[columns_to_scale])
outliers_indices = outliers == -1
dataframe = dataframe[~outliers_indices]

# Create a box plot for each feature
plt.figure(figsize=(12, 6))
sns.boxplot(data=dataframe)
plt.title('Box plot of features')
plt.xticks(rotation=45)
plt.show()

fig, axs = plt.subplots(2, 2, figsize=(10,10), constrained_layout=True)
categorical = ['Age', 'Sex', 'Category']
for i, f in enumerate(categorical):
    sns.countplot(y=f, data=dataframe, ax=axs[i//2][i%2], order=dataframe[f].value_
fig, axs = plt.subplots(3, 4, figsize=(15, 10), constrained_layout=True)
numerical = ['ALB', 'ALP', 'ALB', 'ALT', 'AST', 'BIL', 'CHE', 'CHOL', 'CREA', 'GGT'
for i, f in enumerate(numerical):
    sns.histplot(x=f, data=dataframe, ax=axs[i//4][i%4], bins=100)  # Adjusted inde
plt.show()

dataframe.head(15)
```

```
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\covariance\_robust_
covariance.py:747: UserWarning: The covariance matrix associated to your dataset is
not full rank
  warnings.warn(
```



```
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\categorical.py:641:
FutureWarning: The default of observed=False is deprecated and will be changed to Tr
ue in a future version of pandas. Pass observed=False to retain current behavior or
observed=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\seaborn\_oldcore.py:1119: F
utureWarning: use_inf_as_na option is deprecated and will be removed in a future ver
sion. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```


Age
3
2
4
5
6
1
0
0
25
50
75
100
125
150
175
200
count
Sex
0
1
0
50
100
150
200
250
300
350
count
Category
0
1
0
100
200
300
400
500
count
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0



Out[ ]:

	Category	Age	Sex	ALB	ALP	ALT	AST	BIL	CHE
0	0	2	0	-0.476351	-0.796694	-0.883959	-0.935256	-0.944981	-0.264843
1	0	2	0	-0.476351	-0.708858	-0.708191	-0.918590	-0.979730	0.300867
2	0	2	0	-0.192568	-0.687145	-0.397611	-0.739744	-0.958494	-0.010007
3	0	2	0	-0.317568	-0.799161	-0.493174	-0.932051	-0.834942	-0.211474
4	0	2	0	-0.452703	-0.690106	-0.459044	-0.917949	-0.924710	0.031354
5	0	2	0	-0.371622	-0.842092	-0.699659	-0.950641	-0.898649	0.134089
6	0	2	0	-0.212838	-0.851962	-0.716724	-0.962821	-0.935328	-0.254169
7	0	2	0	-0.351351	-0.849001	-0.404437	-0.877564	-0.861969	-0.412942
8	0	2	0	-0.057432	-0.732544	-0.619454	-0.941026	-0.950772	-0.030020
9	0	2	0	-0.344595	-0.629904	-0.668942	-0.948718	-0.677606	-0.460974
10	0	2	0	-0.280405	-0.797681	-0.645051	-0.933333	-0.851351	-0.635757
11	0	2	0	-0.209459	-0.719220	-0.839590	-0.948718	-0.962355	-0.207472
12	0	2	0	-0.550676	-0.667900	-0.612628	-0.935897	-0.949807	-0.047365
13	0	2	0	-0.459459	-0.800642	-0.744027	-0.923077	-0.951737	-0.327552
14	0	2	0	-0.469595	-0.859363	-0.631399	-0.929487	-0.977799	-0.571714

Train-Test splitting

In [ ]:

```
dataframe_shuffled = dataframe.sample(frac=1)

features, target = dataframe_shuffled.drop('Category', axis=1), dataframe_shuffled[

# Split into training and test set
features_train, features_test, target_train, target_test = train_test_split(
features, target, random_state=0)

# List that contains results of each model
results_list = list()
results_list_cv_only = list()
```

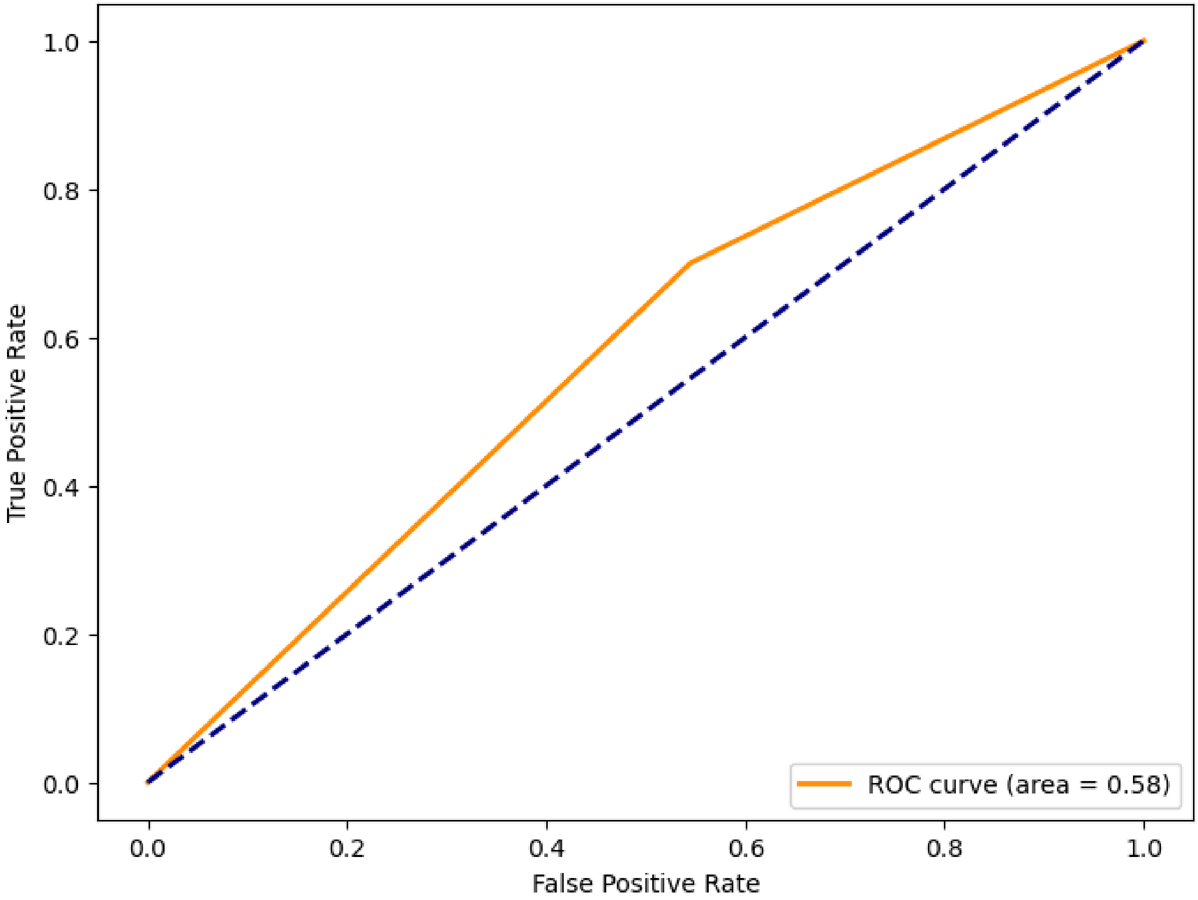
Model testing

Dummy classifier

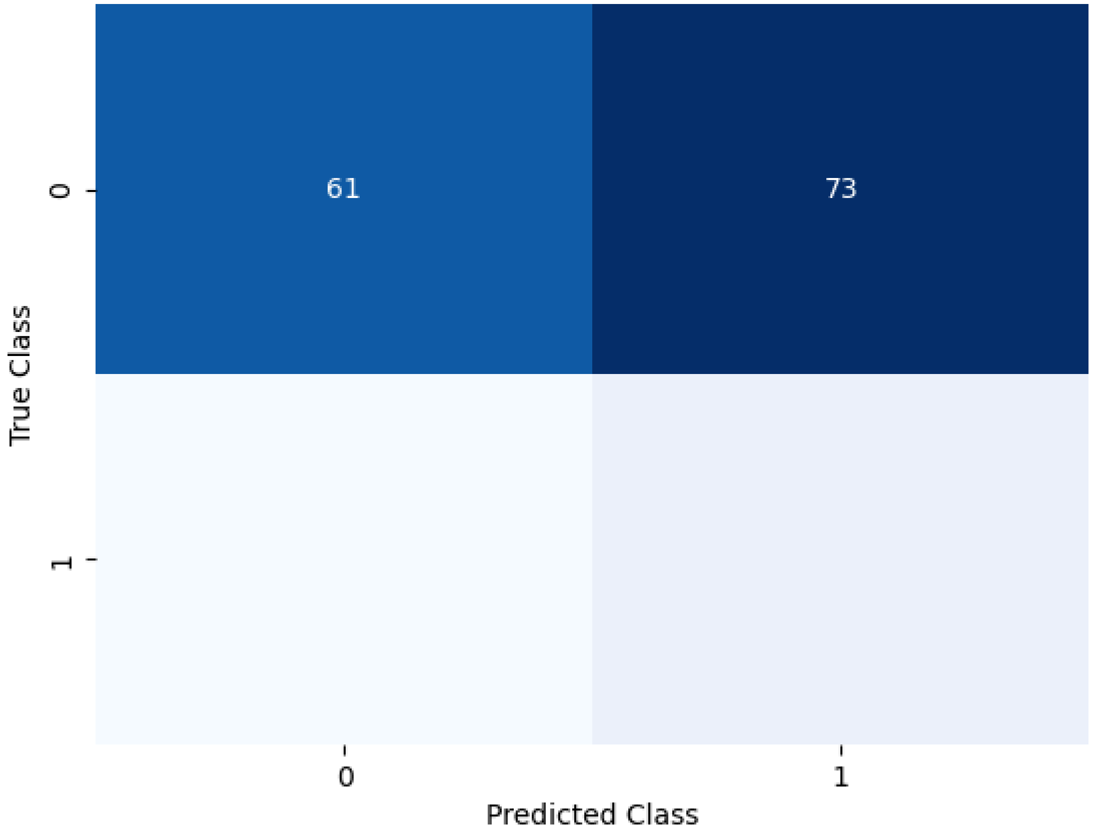
In [ ]:

```
# Create dummy classifier
dummy = DummyClassifier(strategy='uniform', random_state=1)

# "Train" model
dummy.fit(features_train, target_train)

# Predict on test features
dummy_prediction = dummy.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, dummy_prediction, "Dummy classifier")

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, dummy_prediction, "Dummy classifier")

dummy_report = classification_report(target_test, dummy_prediction, output_dict=Tru
print(classification_report(target_test, dummy_prediction))

dummy_bs_report_fromalized = myResultFormalizer(dummy_report, "Dummy classifier")

results_list.append(dummy_bs_report_fromalized)
```

Receiver Operating Characteristic (ROC) - Dummy classifier



Dummy classifier

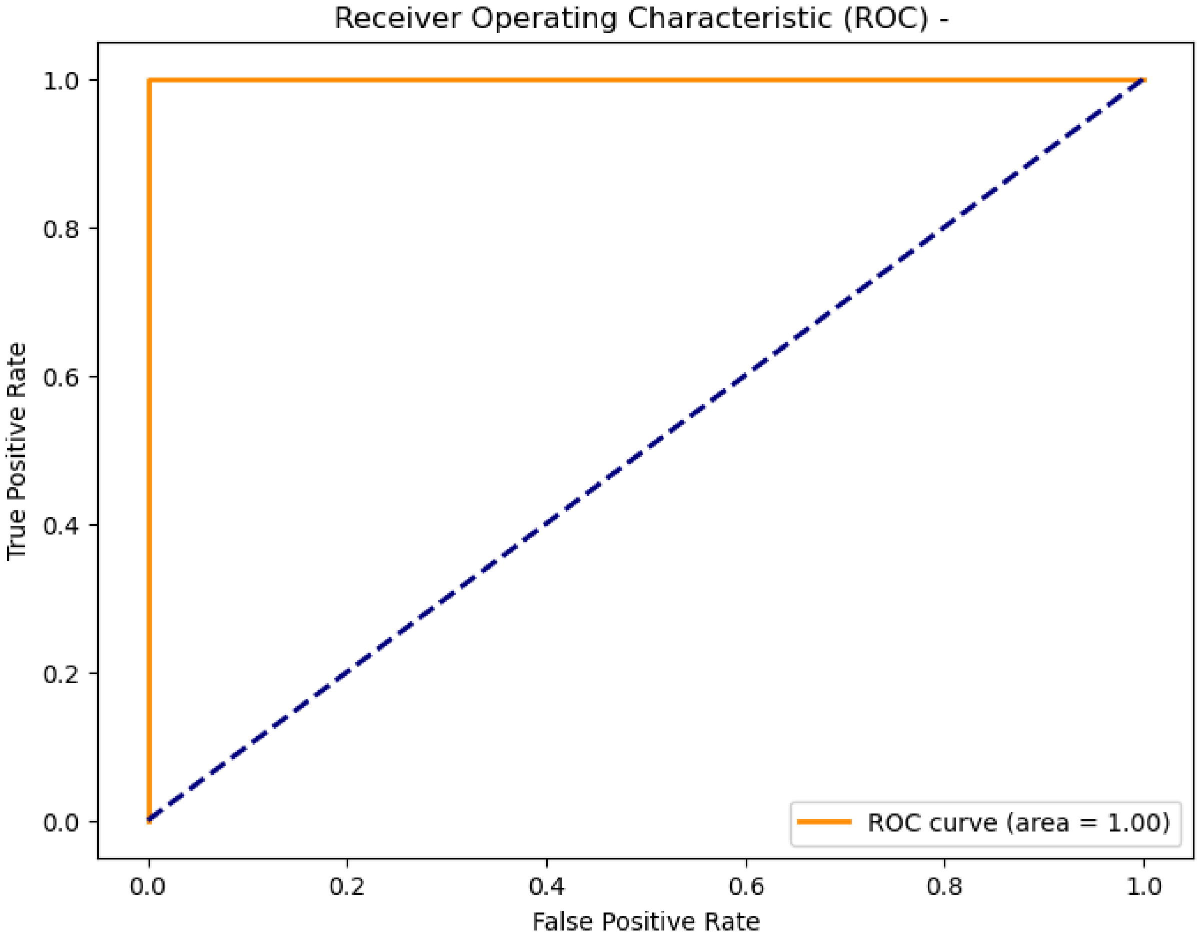


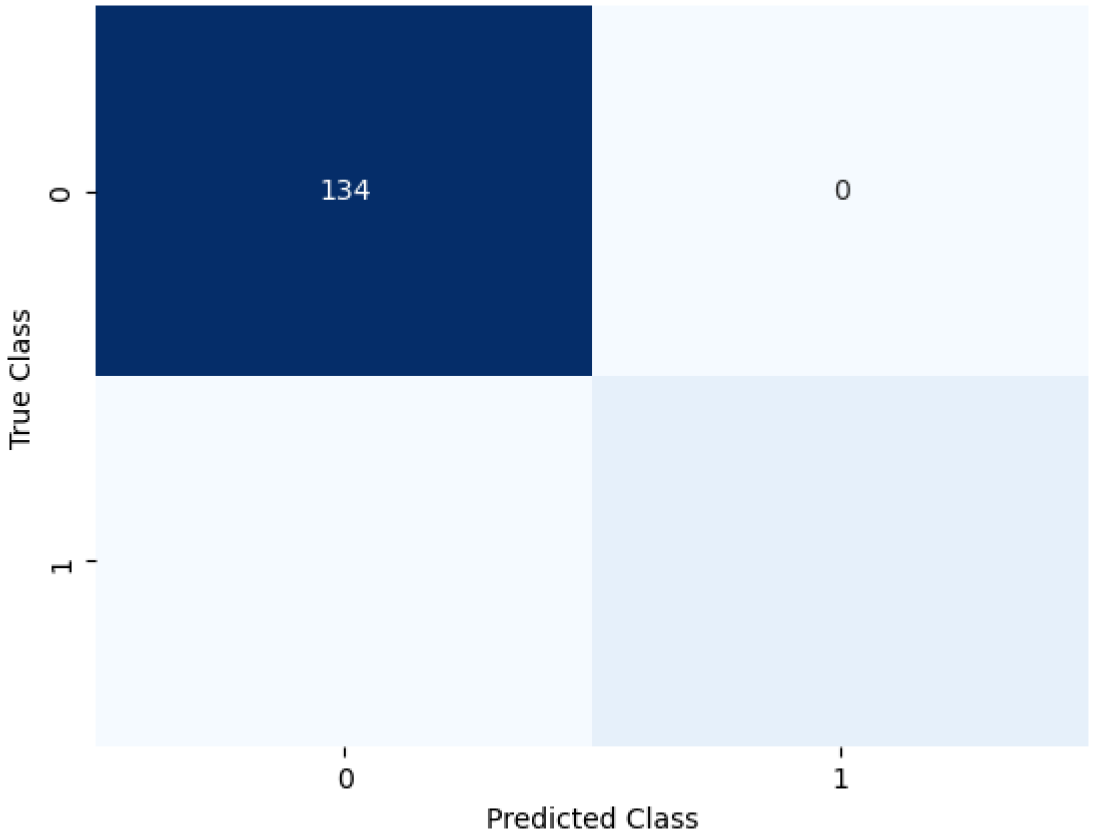
```
              precision    recall  f1-score   support

           0       0.95      0.46      0.62       134
           1       0.09      0.70      0.16        10

    accuracy                           0.47       144
   macro avg       0.52      0.58      0.39       144
weighted avg       0.89      0.47      0.58       144
```

RandomForestClassifier, Basic split

In [ ]:

```
# Create classifier
rf_classifier = RandomForestClassifier()

model_title = "Random Forest Classifier, Basic split"

# Train model
rf_classifier.fit(features_train, target_train)

# Predict on test features
rfc_prediction = rf_classifier.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, rfc_prediction)

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, rfc_prediction)

# Print classification report
print("\nBasic split Random Forest Classifier Classification Report:")
rfc_report = classification_report(target_test, rfc_prediction, output_dict=True)
print(classification_report(target_test, rfc_prediction))

rfc_bs_report_formalized = myResultFormalizer(rfc_report, model_title)

results_list.append(rfc_bs_report_formalized)
```


Receiver Operating Characteristic (ROC) -
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve (area = 1.00)

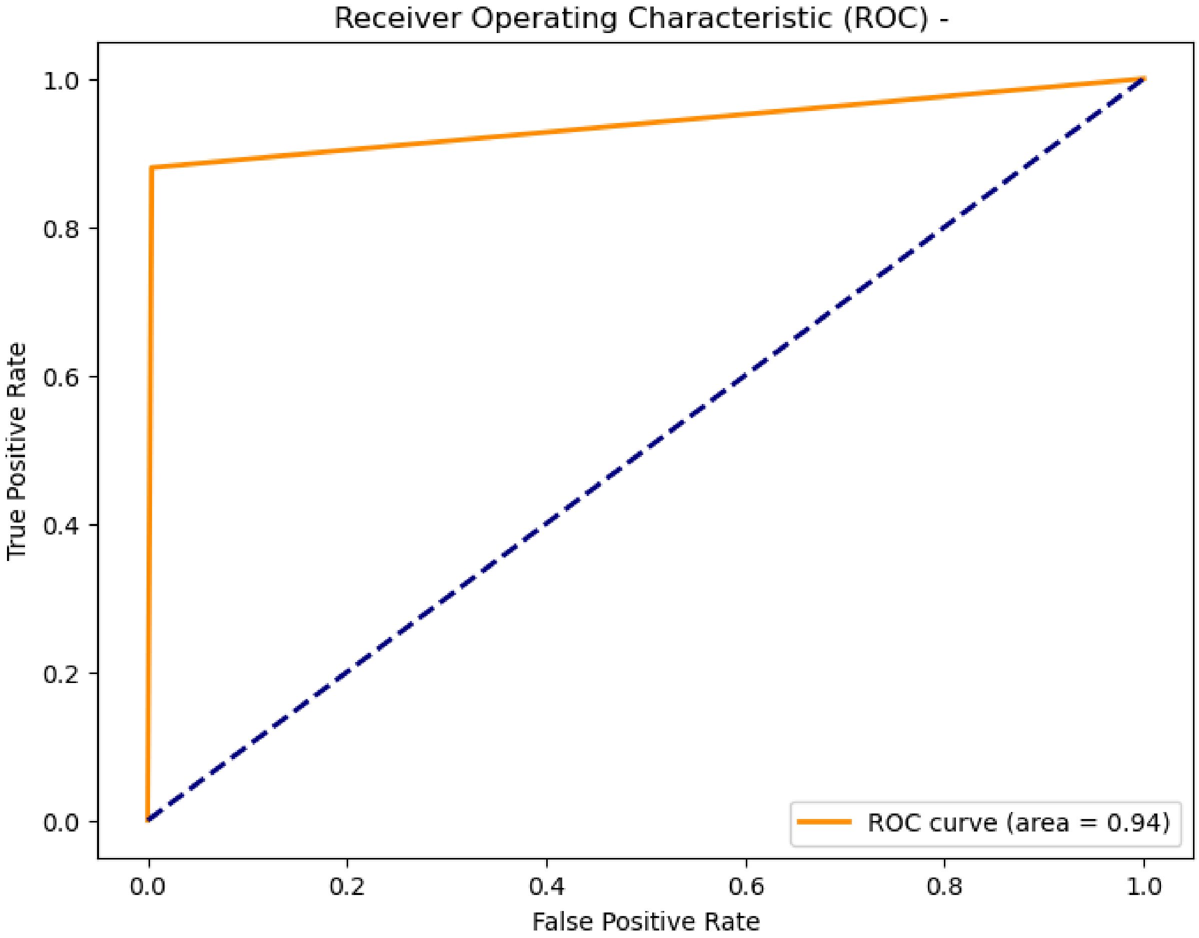

134
0
0
1
True Class
0
1
Predicted Class

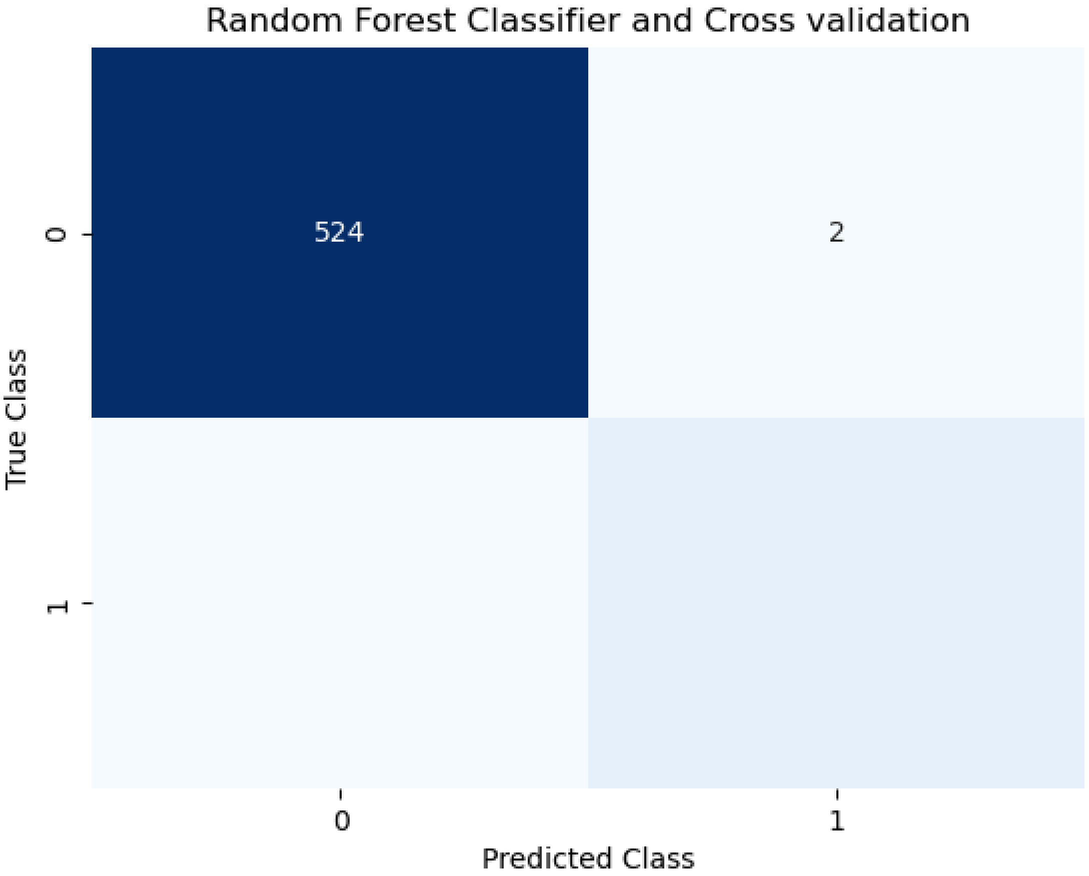
```
Basic split Random Forest Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       134
           1       1.00      1.00      1.00        10

    accuracy                           1.00       144
   macro avg       1.00      1.00      1.00       144
weighted avg       1.00      1.00      1.00       144
```

RandomForestClassifier, Cross-Validation

In [ ]:

```
# Create Random Forest Classifier object
rf_classifier = RandomForestClassifier()

model_title = "Random Forest Classifier and Cross validation"

# Stratified K-Fold cross-validation
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

rf_cv_results, predictions = myCrossValidation(rf_classifier, skf, features, target

myPlotROCcurve(target, predictions)

myPlotConfusionMatrix(target, predictions, model_title)

rf_cv_results.head()

results_list.append(rf_cv_results)
results_list_cv_only.append(rf_cv_results)
```

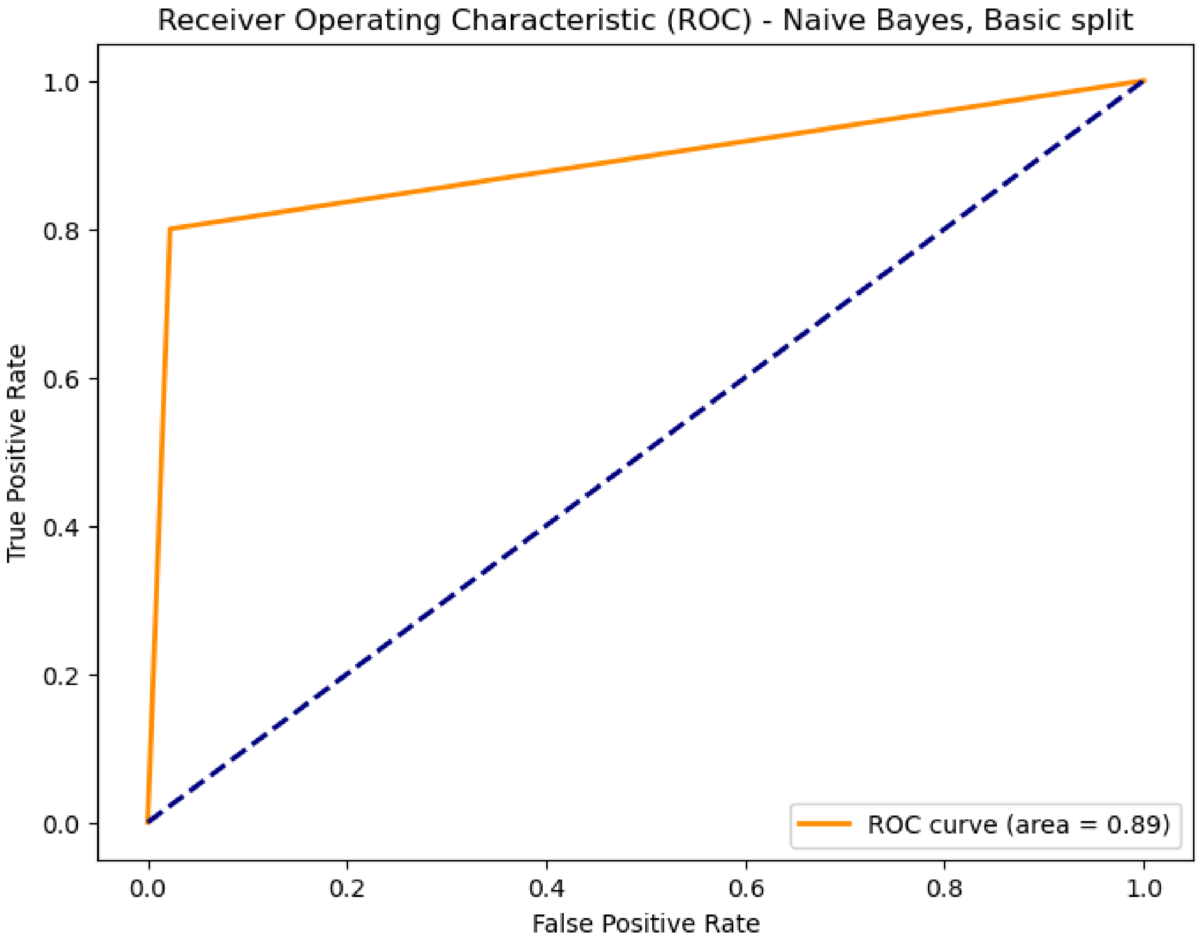

Receiver Operating Characteristic (ROC) -
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
ROC curve (area = 0.94)

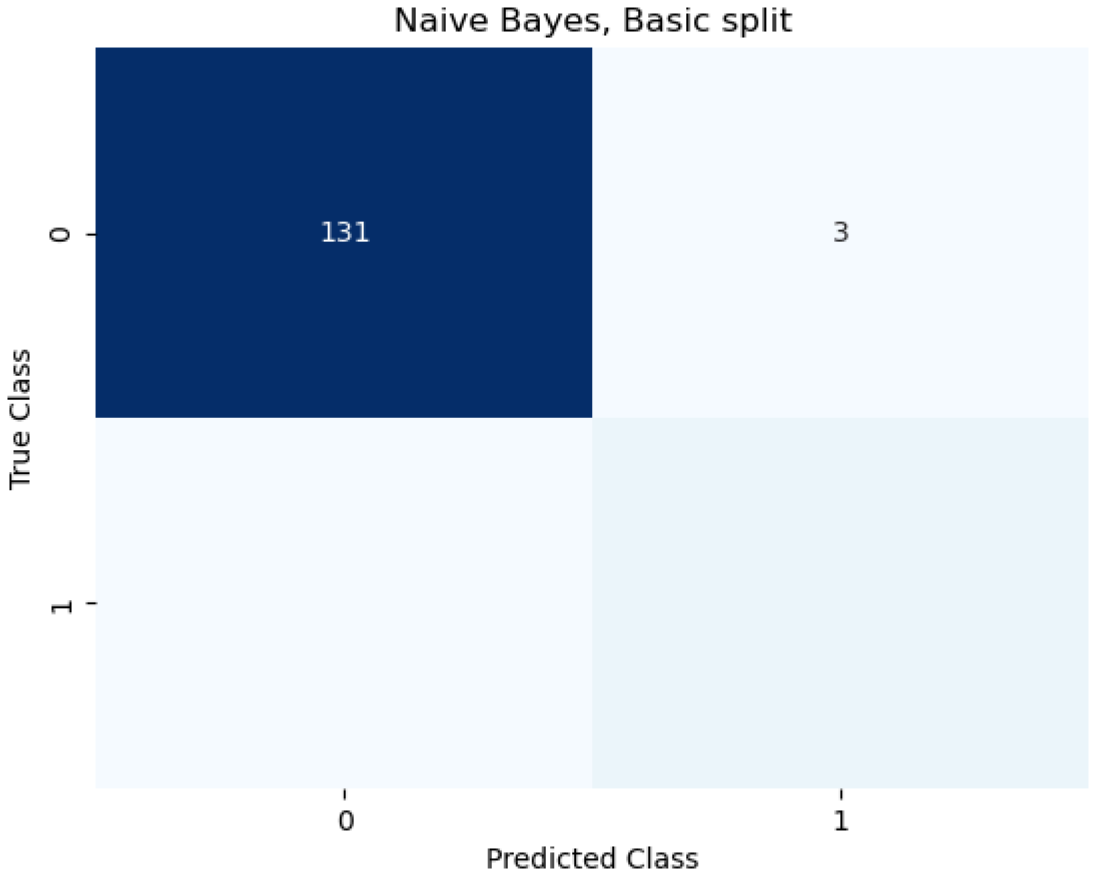

Random Forest Classifier and Cross validation
524
2
0
1
True Class
0
1
Predicted Class

Naive Bayes, Basic split

In [ ]:

```python
# Create Naive Bayes Classifier object
nb_classifier = GaussianNB()

model_title = "Naive Bayes, Basic split"

# Train model
nb_classifier.fit(features_train, target_train)

# Predict on test features
nb_prediction = nb_classifier.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, nb_prediction, model_title)

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, nb_prediction, model_title)

# Print classification report
print("\nBasic split Naive Bayes Classifier Classification Report:")
nb_report = classification_report(target_test, nb_prediction, output_dict=True)
print(classification_report(target_test, nb_prediction))

nb_bs_report_formalized = myResultFormalizer(nb_report, model_title)

results_list.append(nb_bs_report_formalized)
```


Receiver Operating Characteristic (ROC) - Naive Bayes, Basic split
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.89)

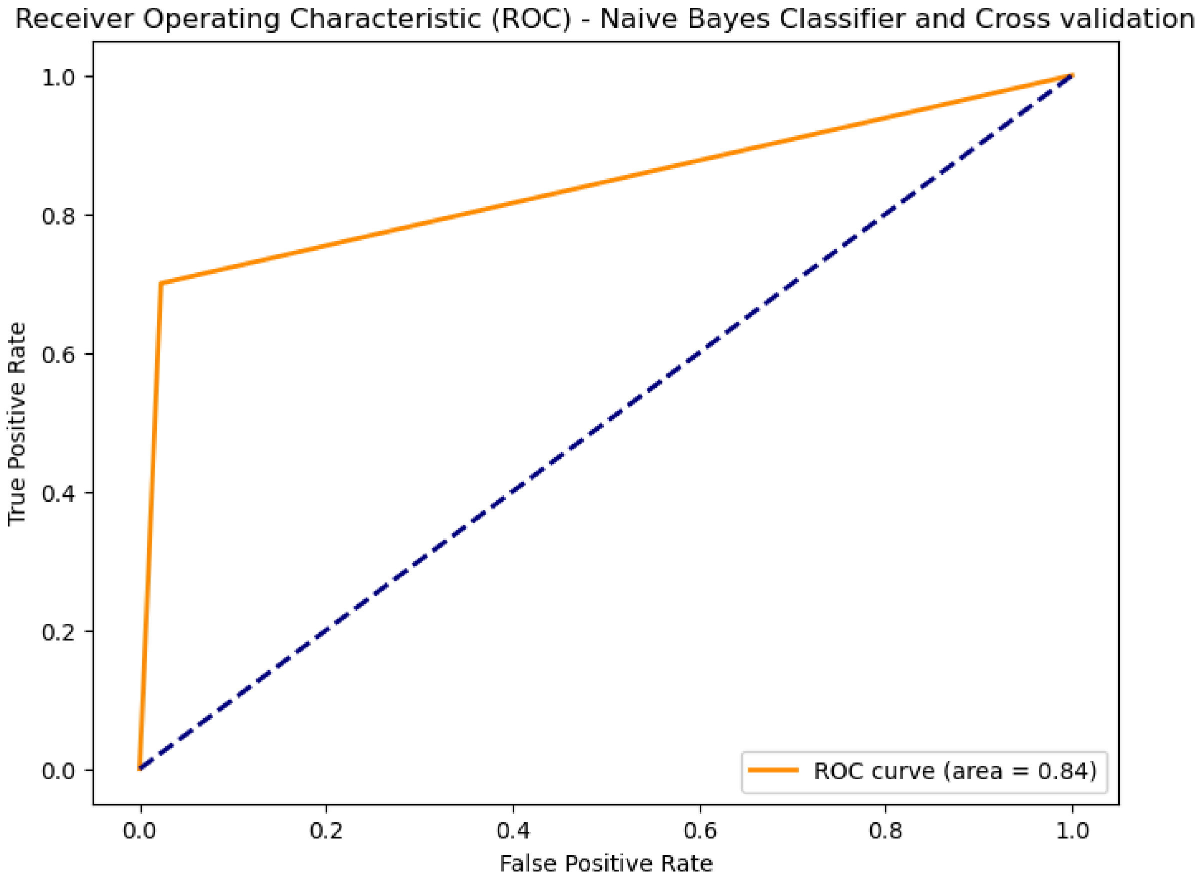

Naive Bayes, Basic split
131
3
True Class
0
1
0
1
Predicted Class

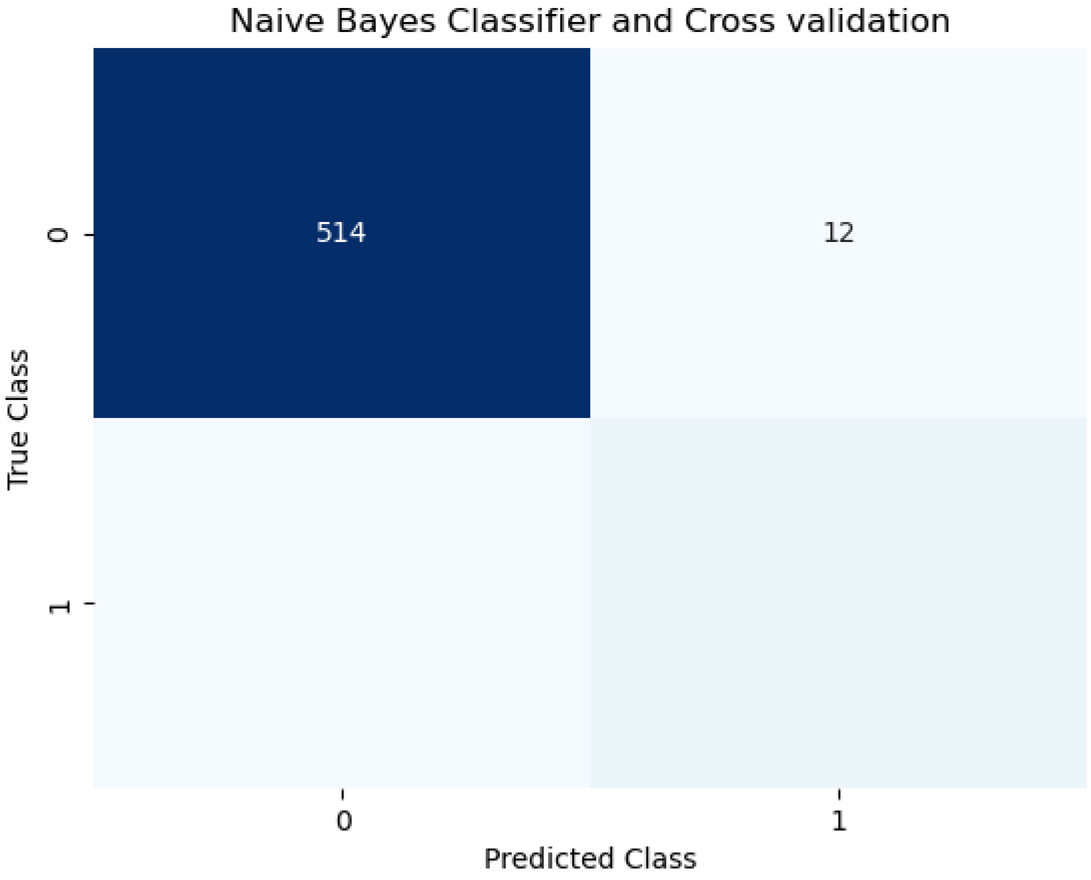
```
Basic split Naive Bayes Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       134
           1       0.73      0.80      0.76        10

    accuracy                           0.97       144
   macro avg       0.86      0.89      0.87       144
weighted avg       0.97      0.97      0.97       144
```

Naive Bayes, Cross-Validation

In [ ]:

```
# Create Naive Bayes Classifier object
nb_classifier = GaussianNB()

model_title = "Naive Bayes Classifier and Cross validation"

# Stratified K-Fold cross-validation
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

nb_cv_results, predictions = myCrossValidation(nb_classifier, skf, features, target

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

nb_cv_results.head()

results_list.append(nb_cv_results)
results_list_cv_only.append(nb_cv_results)
```

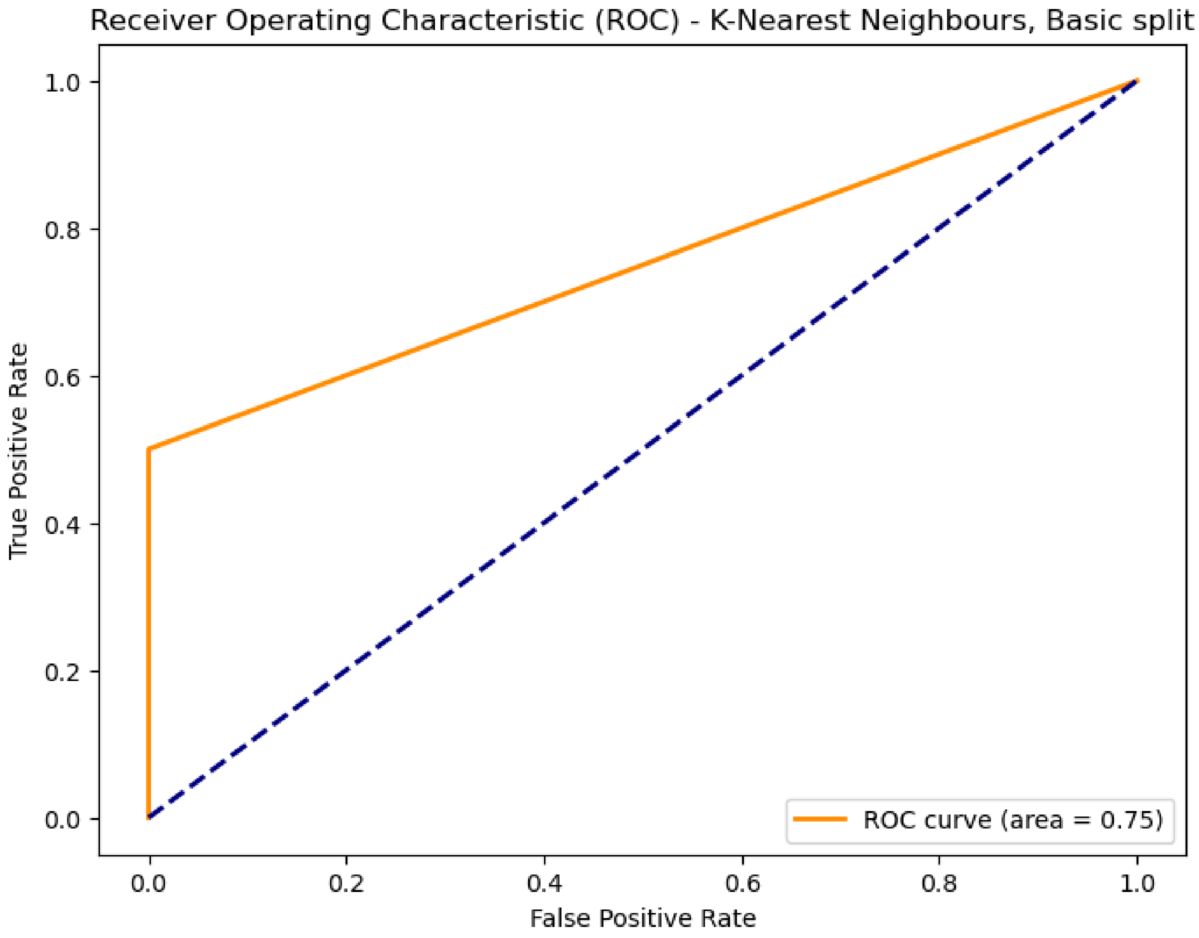


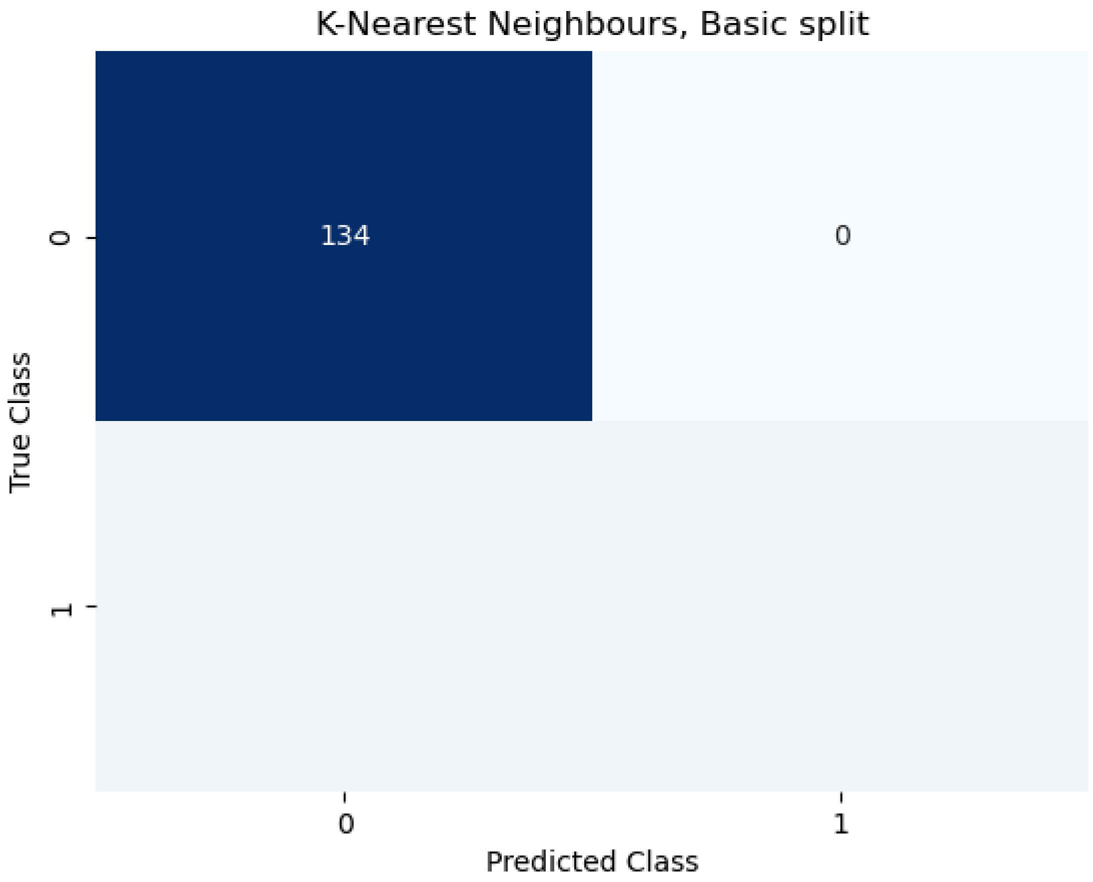


K-Nearest Neighbors, Basic split

In [ ]:

```python
# Create K-Nearest Neighbors classifier object
knn_classifier = KNeighborsClassifier()

model_title = "K-Nearest Neighbours, Basic split"

# Train model
knn_classifier.fit(features_train, target_train)

# Predict on test features
knn_prediction = knn_classifier.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, knn_prediction, model_title)

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, knn_prediction, model_title)

# Print classification report
print("\nBasic split K-Nearest Neighbours Classifier Classification Report:")
knn_bs_report = classification_report(target_test, knn_prediction, output_dict=True
print(classification_report(target_test, knn_prediction))

knn_bs_report_formalized = myResultFormalizer(knn_bs_report, model_title)

results_list.append(knn_bs_report_formalized)
```





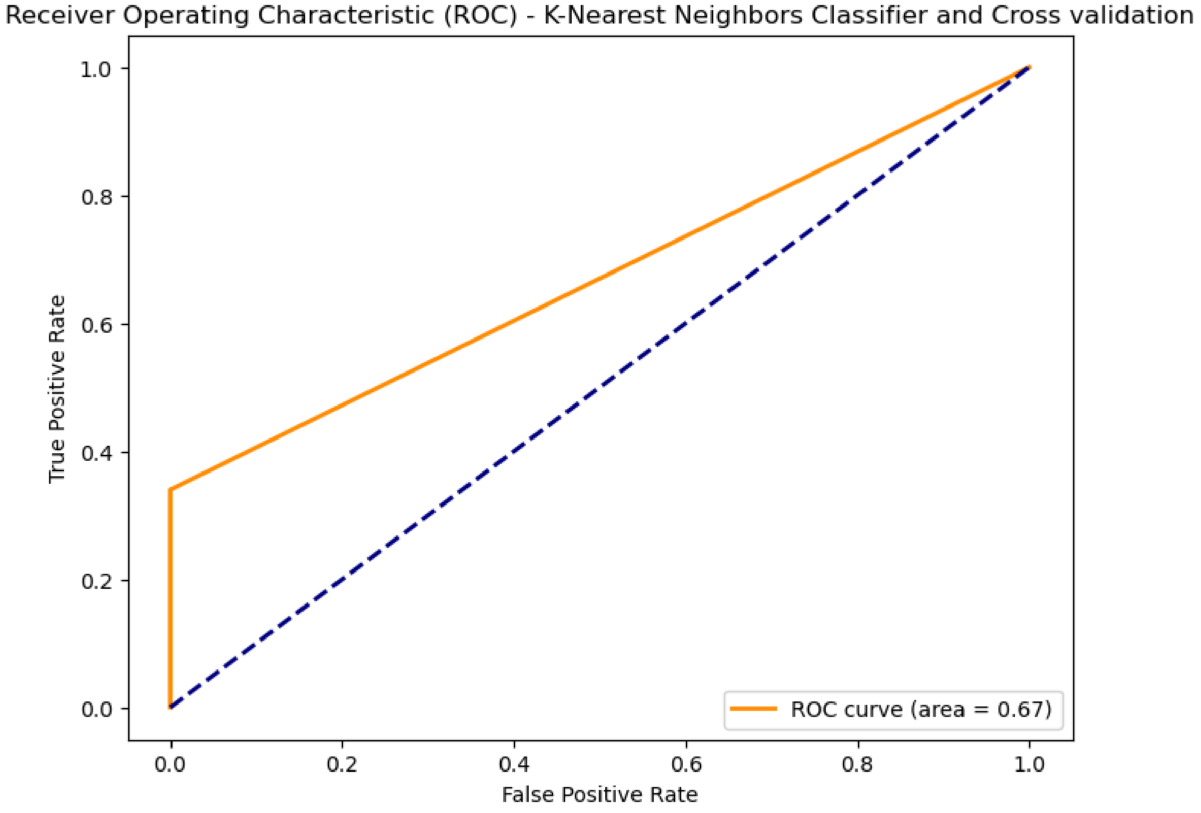
```
Basic split K-Nearest Neighbours Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       134
           1       1.00      0.50      0.67        10

    accuracy                           0.97       144
   macro avg       0.98      0.75      0.82       144
weighted avg       0.97      0.97      0.96       144
```

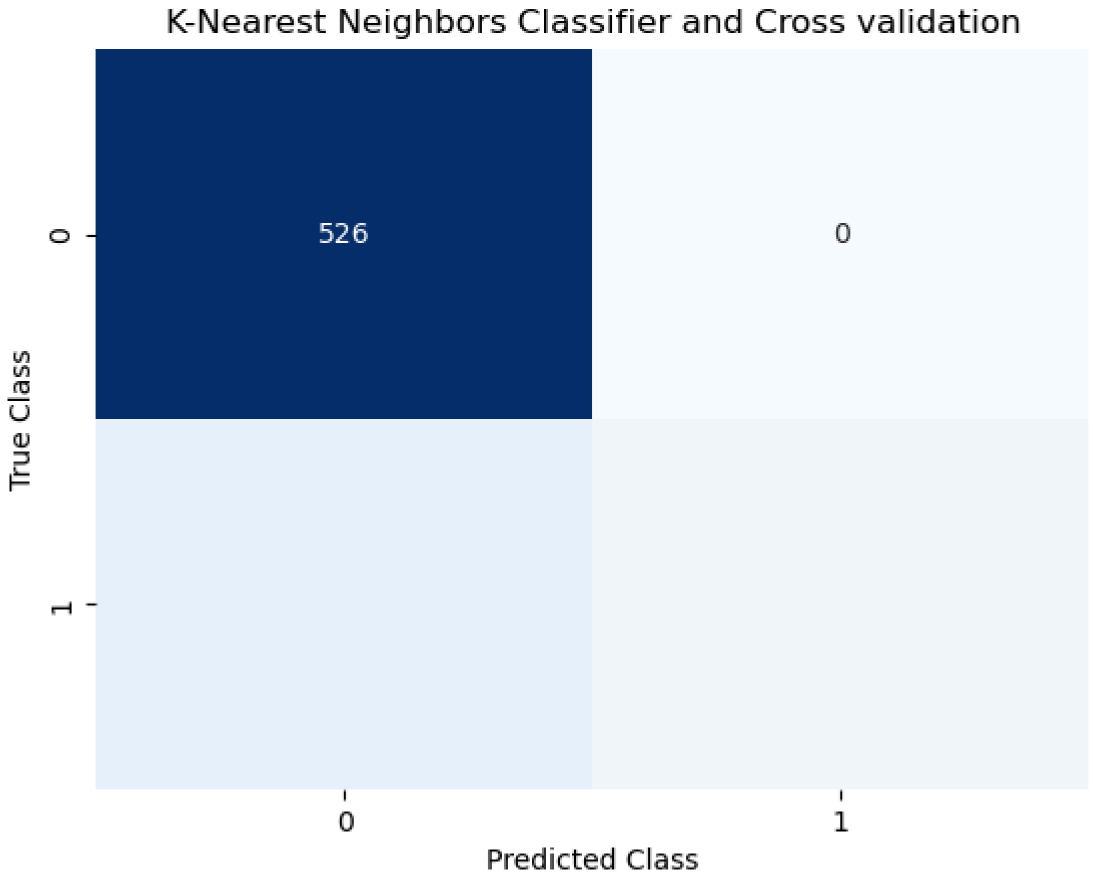
K-Nearest Neighbors, Cross-Validation

In [ ]:

```
# Create K-Nearest Neighbors Classifier object
knn_classifier = KNeighborsClassifier()

model_title = "K-Nearest Neighbors Classifier and Cross validation"

# Stratified K-Fold cross-validation
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

knn_cv_results, predictions = myCrossValidation(knn_classifier, skf, features, targ

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

knn_cv_results.head()
```

```
results_list.append(knn_cv_results)
results_list_cv_only.append(knn_cv_results)
```

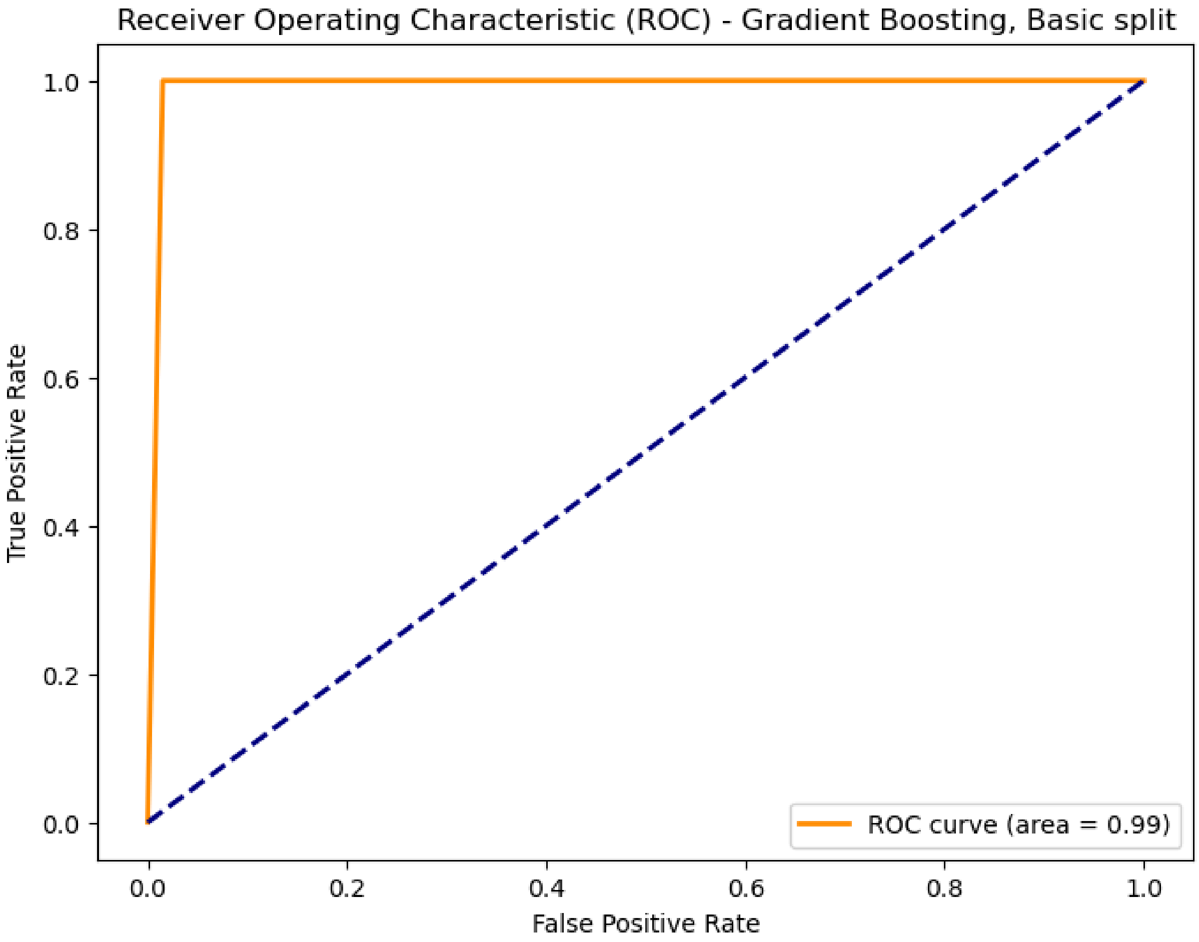


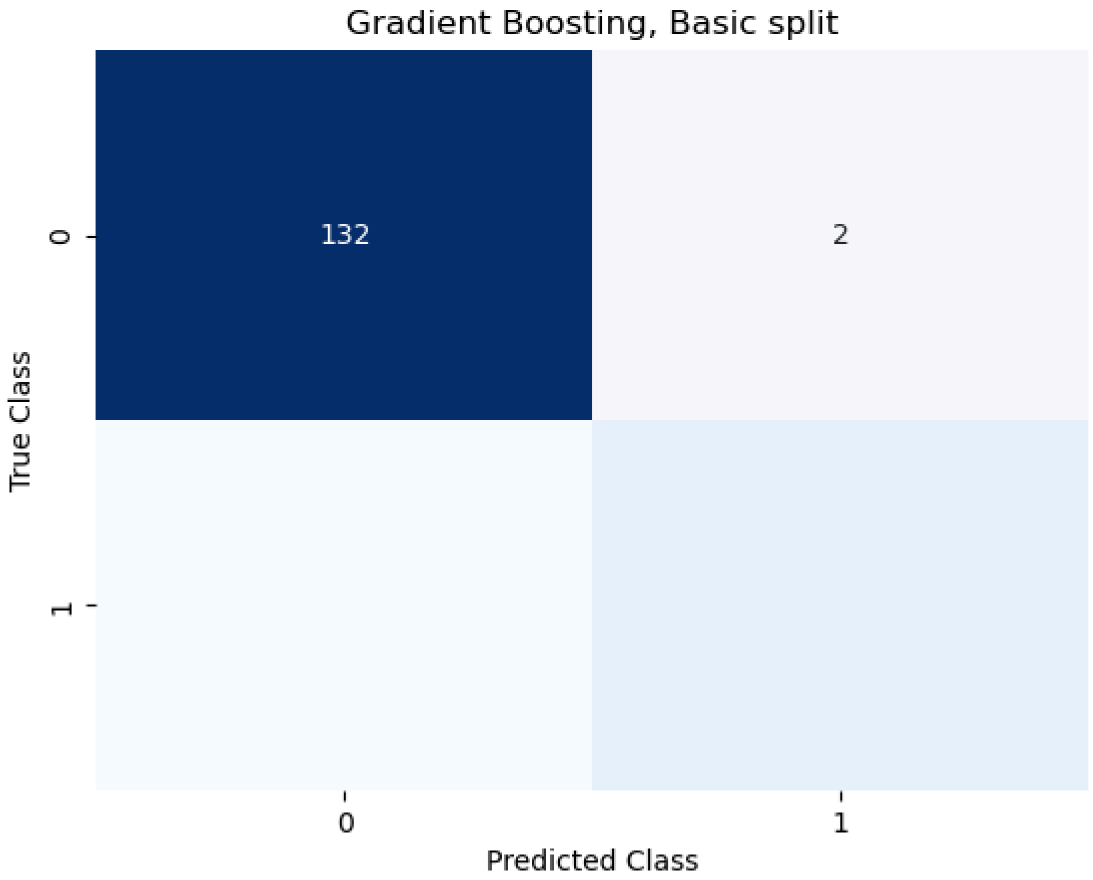


Gradient Boosting, Basic split

In [ ]:

```
# Create Gradient Boosting classifier object
gb_classifier = GradientBoostingClassifier()

model_title = "Gradient Boosting, Basic split"

# Train model
gb_classifier.fit(features_train, target_train)

# Predict on test features
gb_prediction = gb_classifier.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, gb_prediction, model_title)

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, gb_prediction, model_title)

# Print classification report
print("\nBasic split Gradient Boosting Classifier Classification Report:")
gb_bs_report = classification_report(target_test, gb_prediction, output_dict=True)
print(classification_report(target_test, gb_prediction))

gb_bs_report_formalized = myResultFormalizer(gb_bs_report, model_title)

results_list.append(gb_bs_report_formalized)
```


Receiver Operating Characteristic (ROC) - Gradient Boosting, Basic split
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.99)

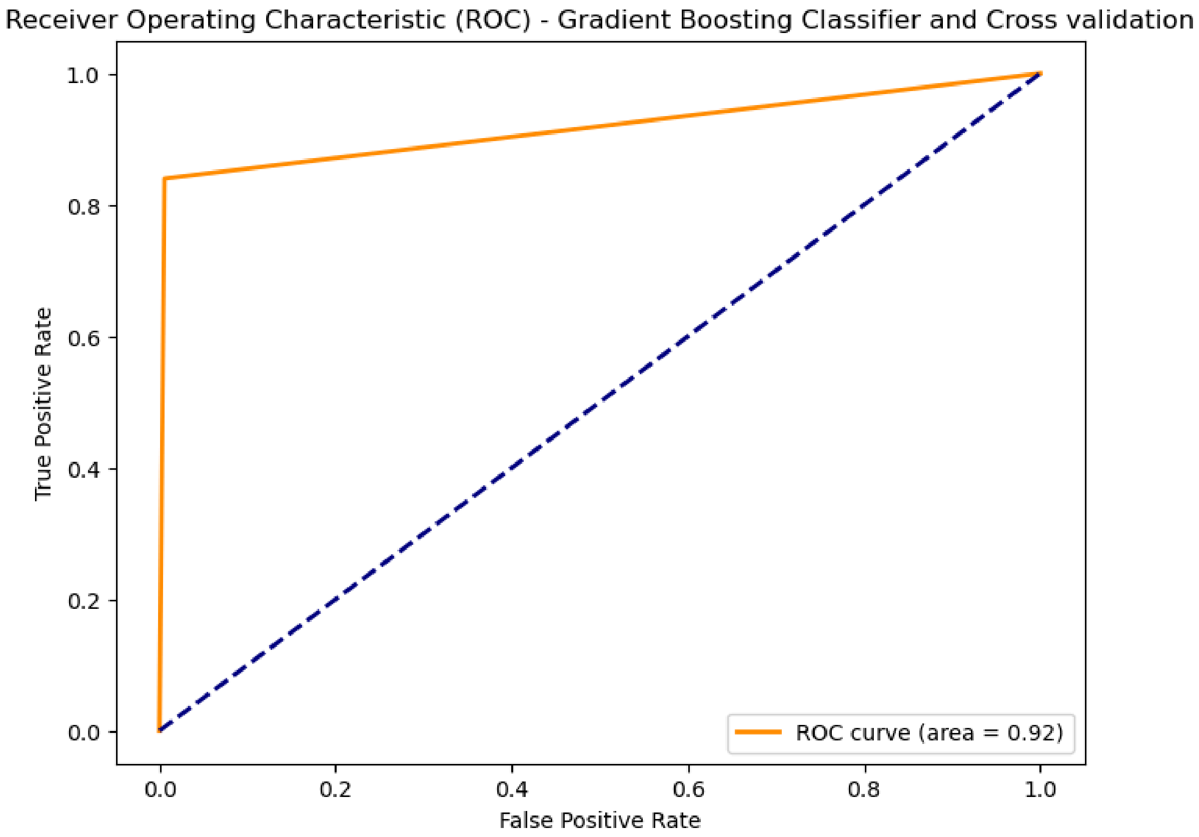

Gradient Boosting, Basic split
True Class
Predicted Class
0
1
132
2

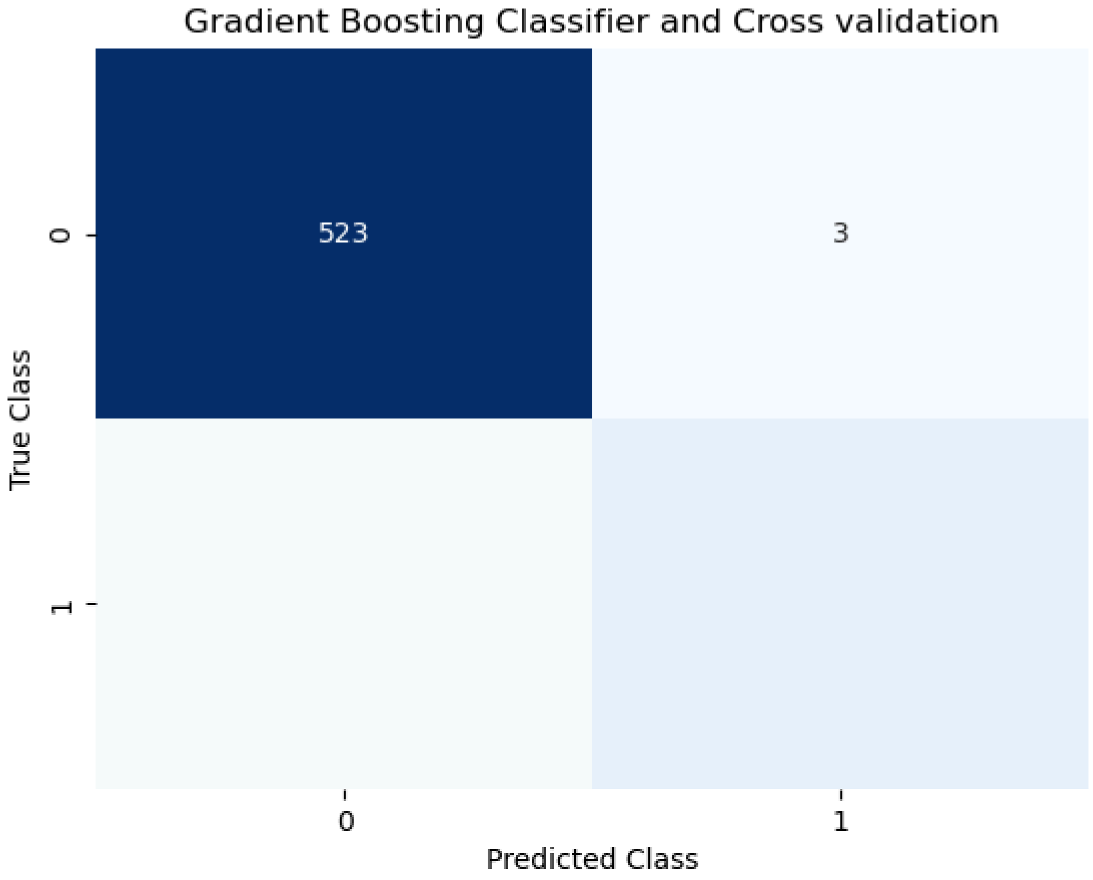
```
Basic split Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.99      0.99       134
           1       0.83      1.00      0.91        10

    accuracy                           0.99       144
   macro avg       0.92      0.99      0.95       144
weighted avg       0.99      0.99      0.99       144
```

Gradient Boosting, Cross-Validation

In [ ]:

```
# Create Gradient Boosting Classifier object
gb_classifier = GradientBoostingClassifier()

model_title = "Gradient Boosting Classifier and Cross validation"

# Stratified K-Fold cross-validation
skf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1)

gb_cv_results, predictions = myCrossValidation(gb_classifier, skf, features, target

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

gb_cv_results.head()

results_list.append(gb_cv_results)
results_list_cv_only.append(gb_cv_results)
```

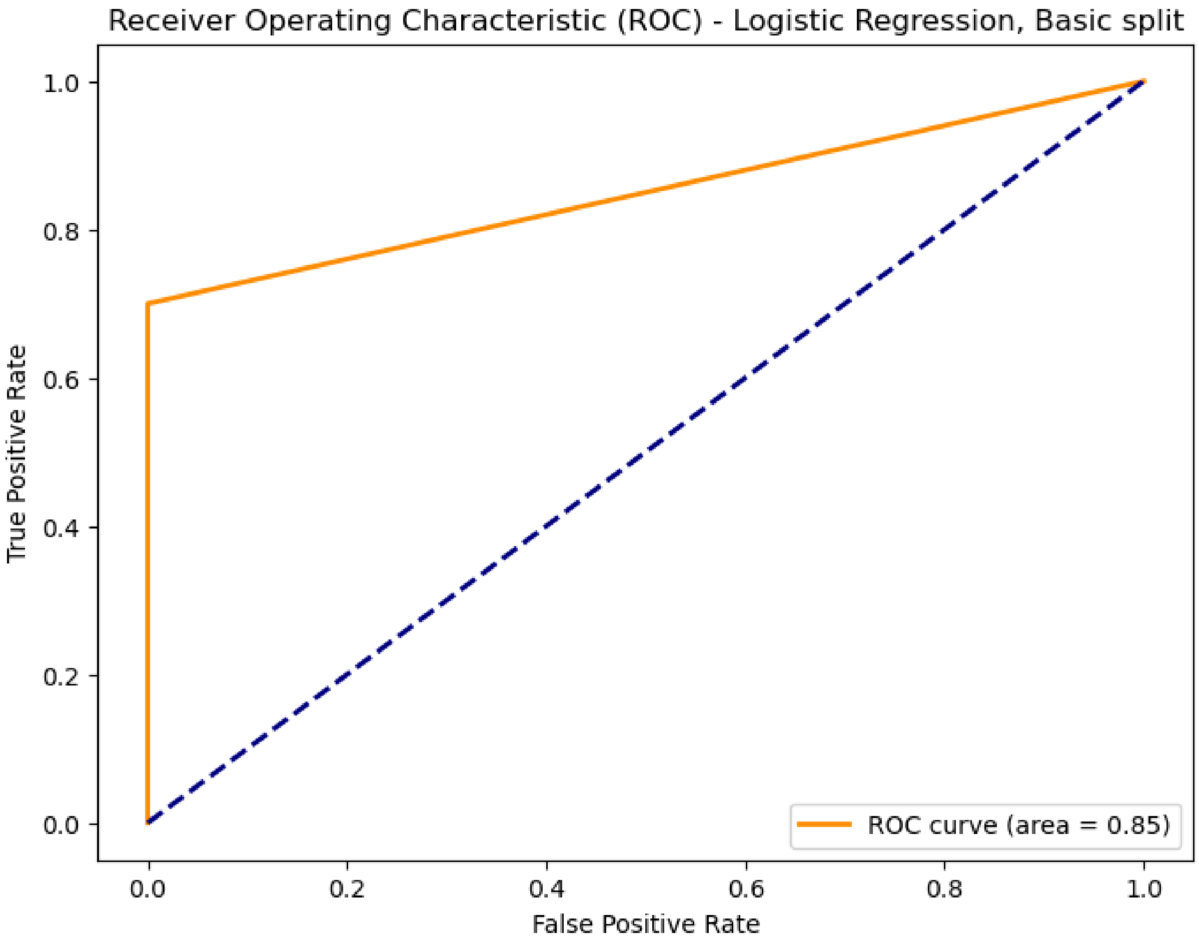


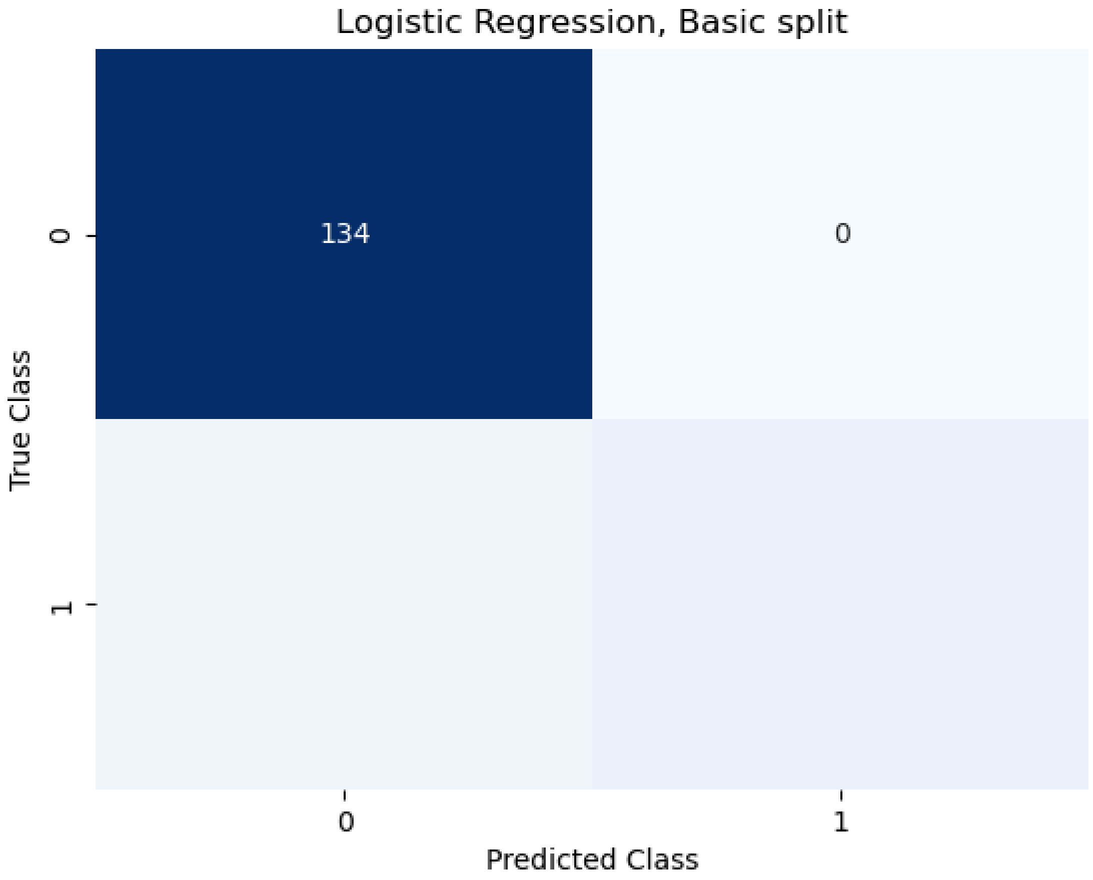


Logistic Regression, Basic split

In [ ]:

```python
# Create Logistic Regression classifier object
lr_classifier = LogisticRegression()

model_title = "Logistic Regression, Basic split"

# Train model
lr_classifier.fit(features_train, target_train)

# Predict on test features
lr_prediction = lr_classifier.predict(features_test)

# Visualize the ROC curve
myPlotROCcurve(target_test, lr_prediction, model_title)

# Visualize the confusion matrix
myPlotConfusionMatrix(target_test, lr_prediction, model_title)

# Print classification report
print("\nBasic split Logistic Regression Classifier Classification Report:")
lr_bs_report = classification_report(target_test, lr_prediction, output_dict=True)
print(classification_report(target_test, lr_prediction))

gb_bs_report_formalized = myResultFormalizer(lr_bs_report, model_title)

results_list.append(gb_bs_report_formalized)
```



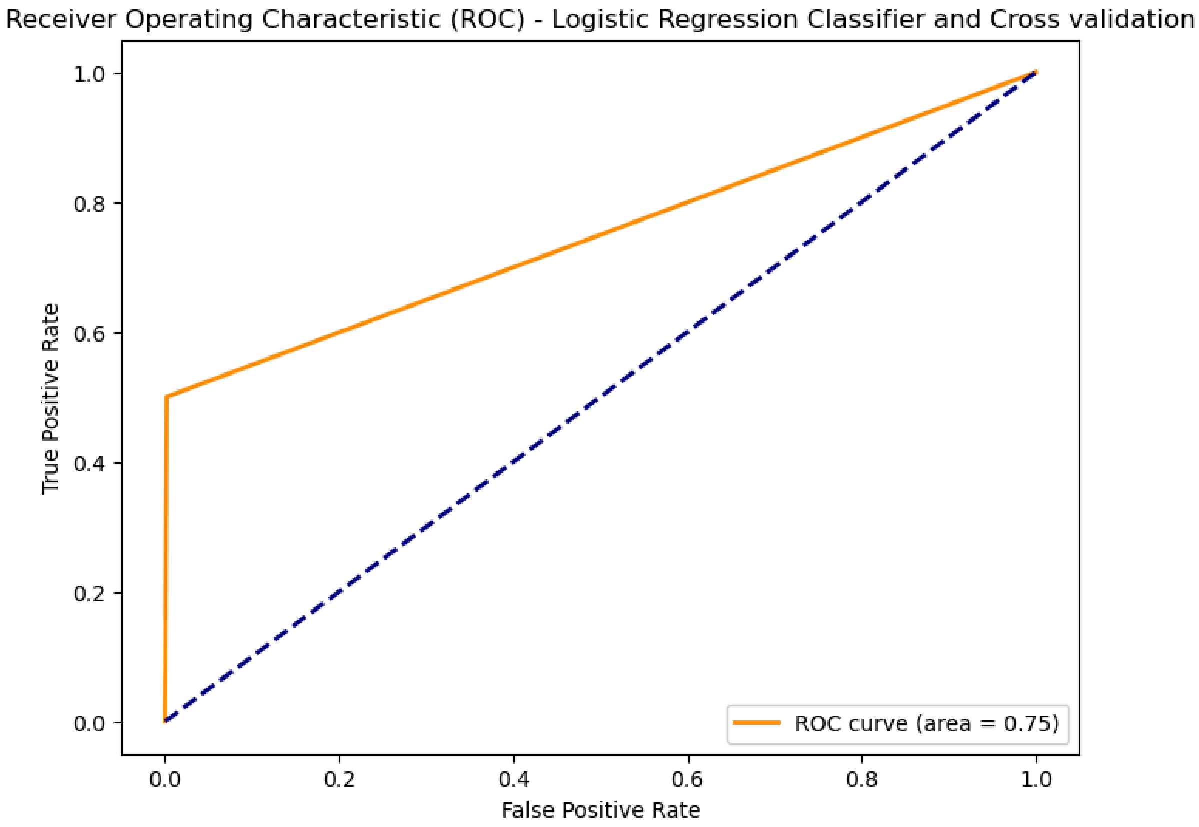


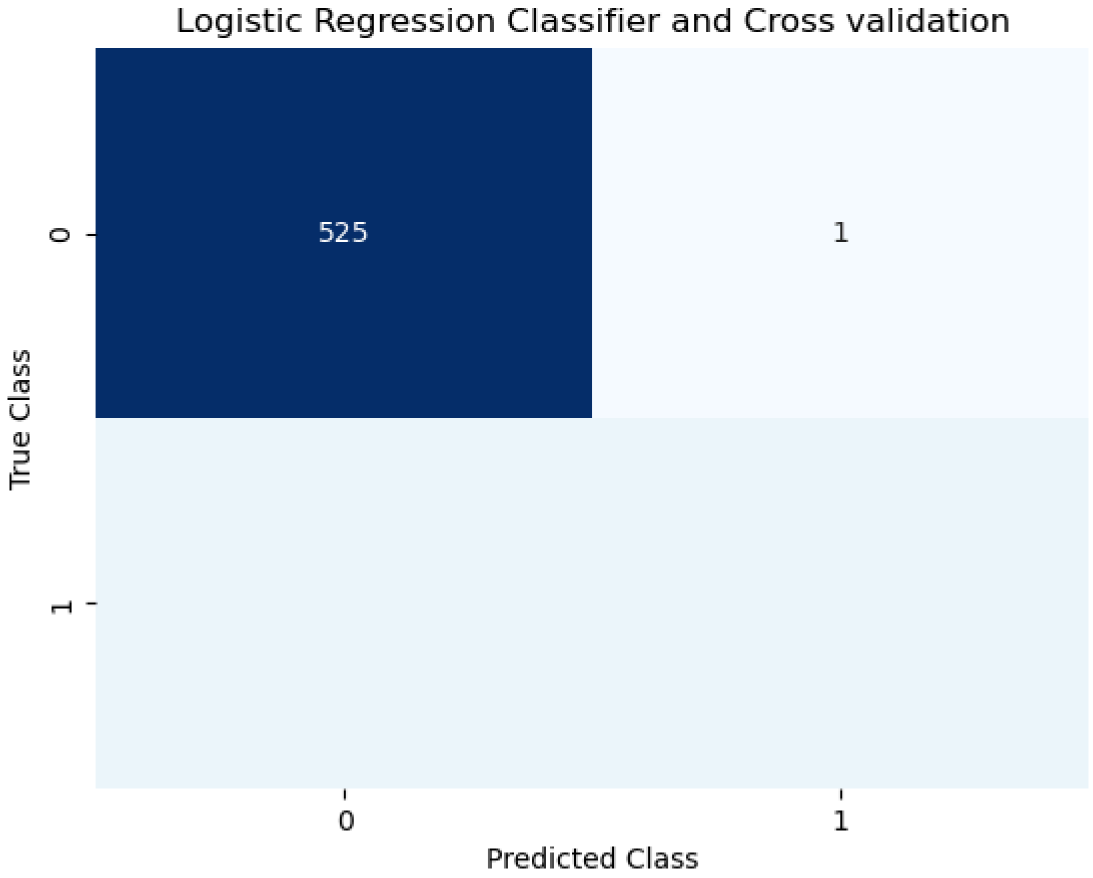
```
Basic split Logistic Regression Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       134
           1       1.00      0.70      0.82        10

    accuracy                           0.98       144
   macro avg       0.99      0.85      0.91       144
weighted avg       0.98      0.98      0.98       144
```

Logistic Regression, Cross-Validation

In [ ]:

```
# Create Logistic Regression classifier object
lr_classifier_cv = LogisticRegression()

model_title = "Logistic Regression Classifier and Cross validation"

lr_cv_results, predictions = myCrossValidation(lr_classifier_cv, skf, features, tar

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

lr_cv_results.head()

results_list.append(lr_cv_results)
results_list_cv_only.append(lr_cv_results)
```

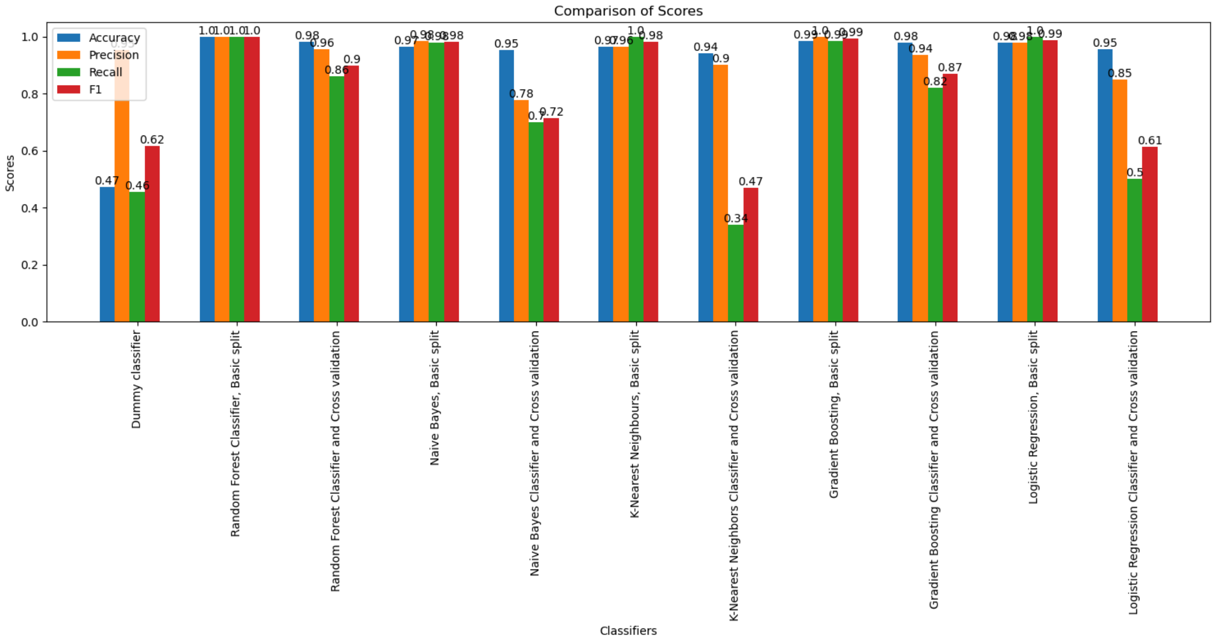


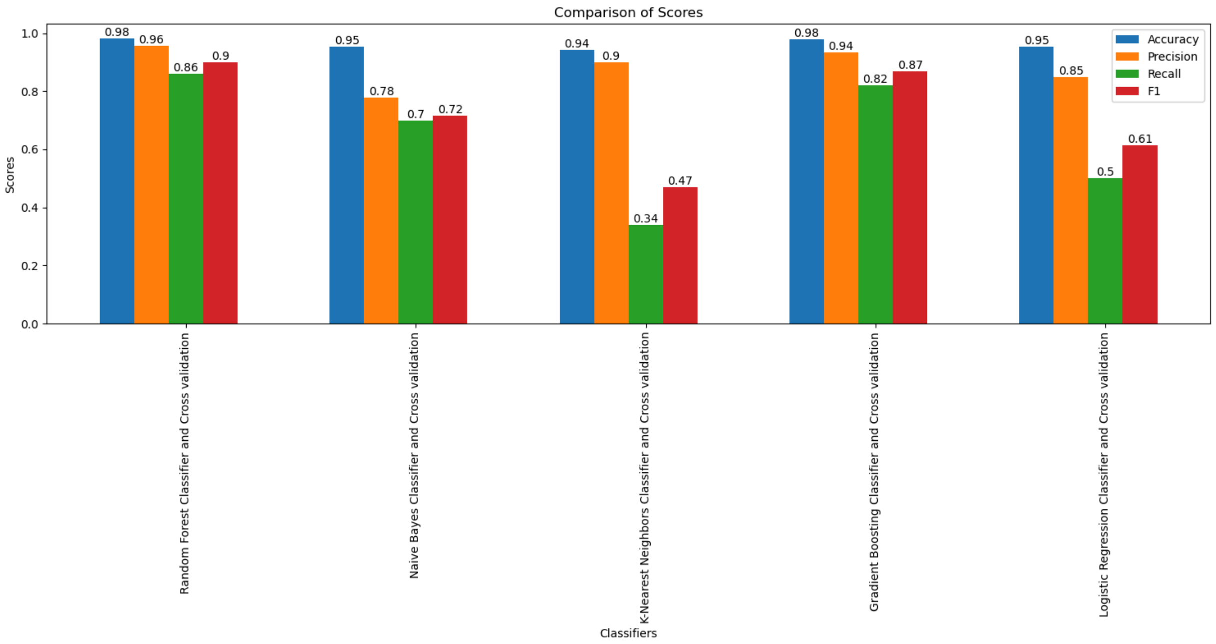


Results compared

In [ ]:

```
def plot_classifier_scores(dataframes, metrics, angle = 90):
    # Plotting
    bar_width = 0.15
    index = np.arange(len(dataframes))

    fig, ax = plt.subplots(figsize=(15, 8))

    for i, metric in enumerate(metrics):
        scores = [df[metric].iloc[0] for df in dataframes]
        bars = ax.bar(index + (i - len(metrics) / 2) * bar_width, scores, bar_width

        # Add value labels on top of each bar
        for bar in bars:
            height = bar.get_height()
            ax.text(bar.get_x() + bar.get_width()/2, height, round(height, 2), ha='

    ax.set_xlabel('Classifiers')
    ax.set_ylabel('Scores')
    ax.set_title('Comparison of Scores')
    ax.set_xticks(index)
    ax.set_xticklabels([df['Model'].iloc[0] for df in dataframes], rotation=angle)
    ax.legend()

    plt.tight_layout()
    plt.show()



# List of feature names
feature_names = ["Accuracy", "Precision", "Recall", "F1"]


plot_classifier_scores(results_list, feature_names)

plot_classifier_scores(results_list_cv_only, feature_names)
```





Hiperparameters tuning

Nećemo razmatrati rezultate dobijene klasičnom podelom na train i test skupove zato što je dataset jako nebalansiran i rezultati koji su dobijeni nisu od relevantne koristi.

Za dalje podešavanje hiperparametara nastavljamo sa Random Forest i Gradient Boosting klasifikatorima zato što su dali najbolje rezultate.

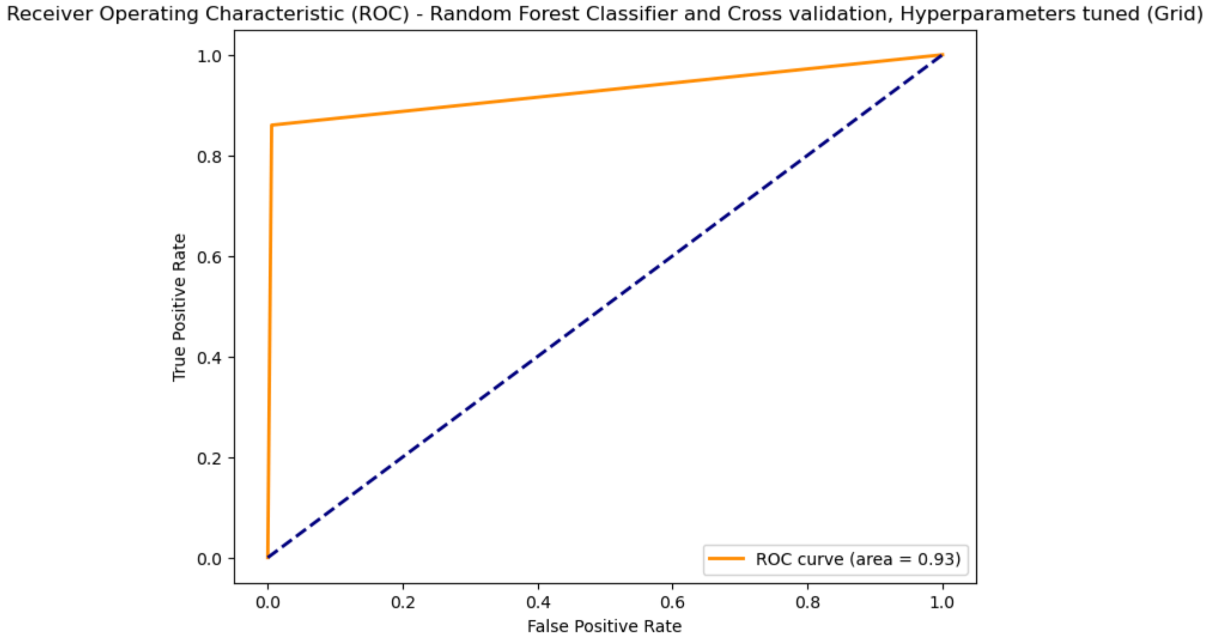
Random Forest Hiperparameters tuning

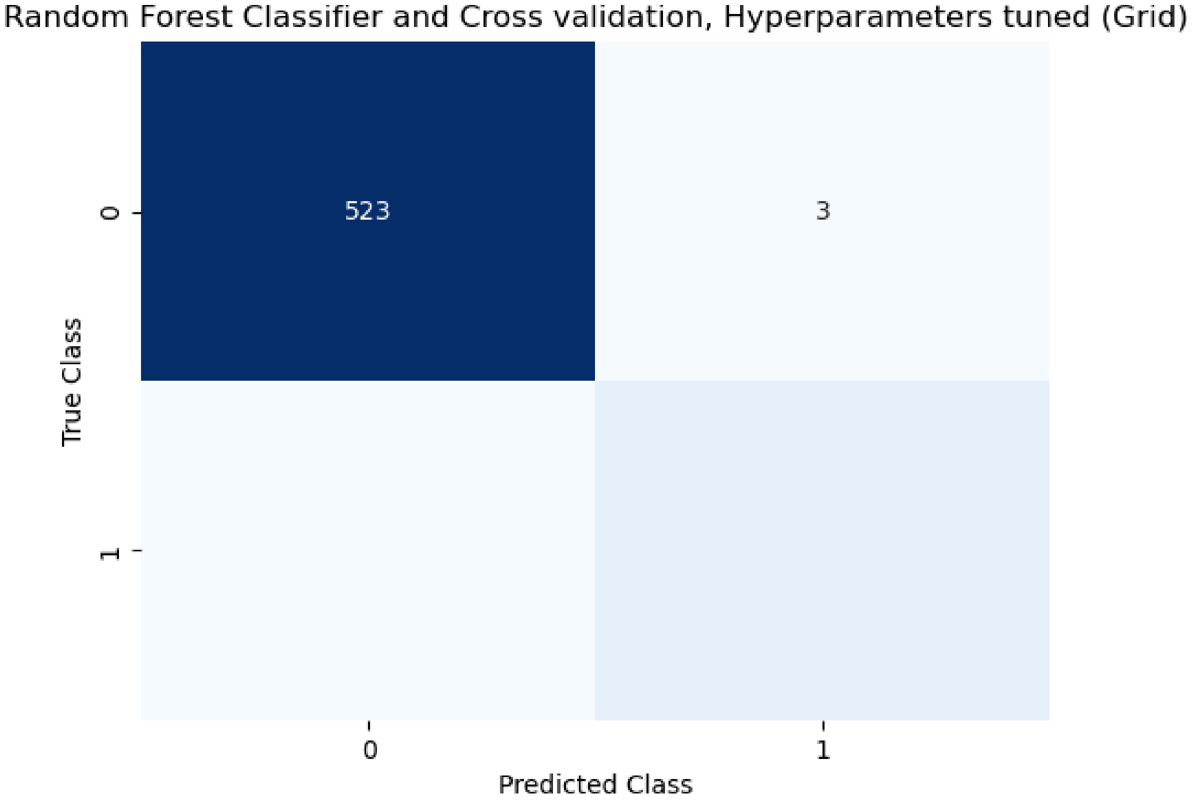
In [ ]:

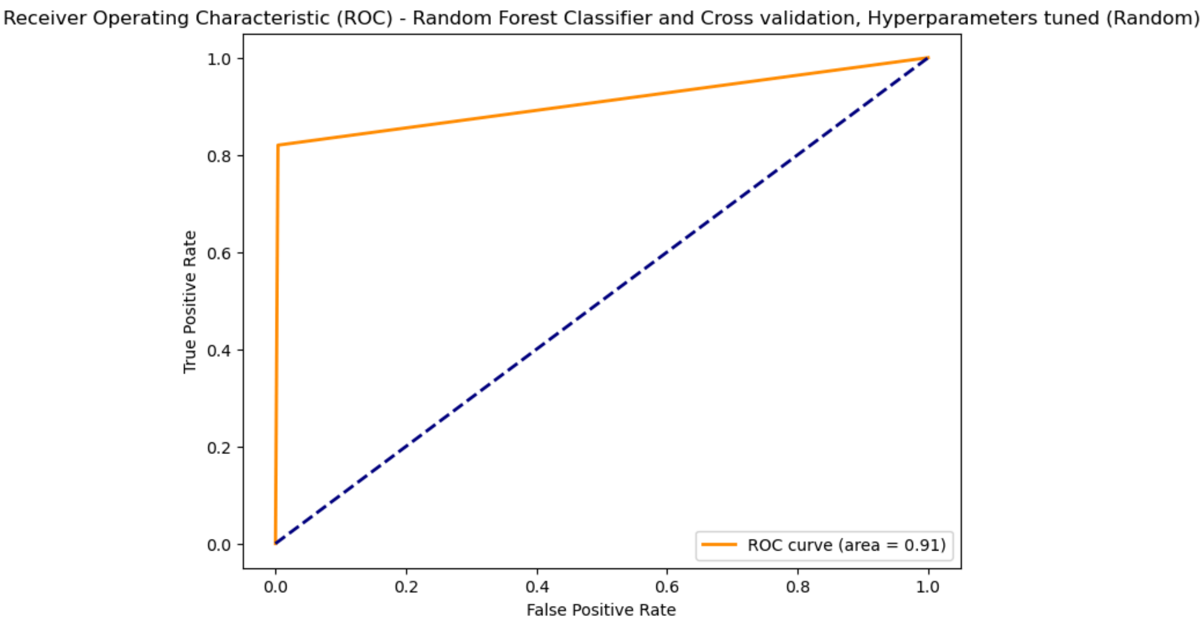
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

# Create a Random Forest classifier object
rf_classifier = RandomForestClassifier()

# Define the hyperparameter grid
param_grid = {
    'n_estimators': [10, 50, 100, 200],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2', None],
    'bootstrap': [True, False]
}

# Perform randoized search
np.random.seed(42)
random_search = RandomizedSearchCV(estimator=rf_classifier, param_distributions=par
random_search.fit(features, target)

rand_params = random_search.best_params_
```

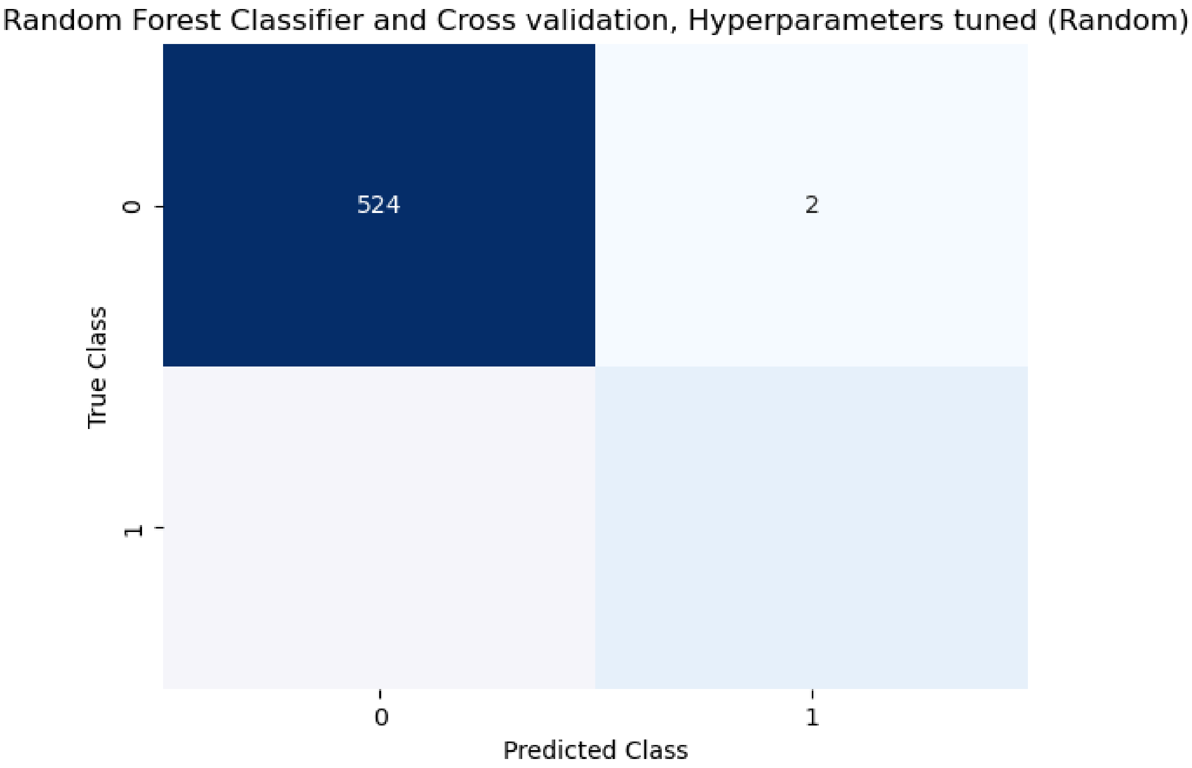
```
## Perform grid search
#grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, n
#grid_search.fit(features, target)
## Best hyperparameters: {'bootstrap': True, 'max_depth': 20, 'max_features': 'log2

## Get the best hyperparameters
# best_params = grid_search.best_params_
# print("Best hyperparameters:", best_params)

best_params = {'bootstrap': True,
               'max_depth': 20,
               'max_features': 'log2',
               'min_samples_leaf': 1,
               'min_samples_split': 5,
               'n_estimators': 50}

best_params_classifier = RandomForestClassifier(**best_params)

model_title = "Random Forest Classifier and Cross validation, Hyperparameters tuned

rf_cv_hp_grid_results, predictions = myCrossValidation(best_params_classifier, skf,

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

rf_cv_hp_grid_results.head()


rnd_params_classifier = RandomForestClassifier(**rand_params)

model_title = "Random Forest Classifier and Cross validation, Hyperparameters tuned

rf_cv_hp_rnd_results, predictions = myCrossValidation(rnd_params_classifier, skf, f

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

rf_cv_hp_rnd_results.head()


final_results = list()
final_results.append(rf_cv_hp_grid_results)
final_results.append(rf_cv_hp_rnd_results)
```

```
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selection\_va
lidation.py:425: FitFailedWarning:
5 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score
='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
4 fits failed with the following error:
Traceback (most recent call last):
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selec
tion\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 1144, in wrapper
    estimator._validate_params()
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 637, in _validate_params
    validate_parameter_constraints(
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\utils\_para
m_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter
of RandomForestClassifier must be an int in the range [1, inf), a float in the range
(0.0, 1.0], a str among {'sqrt', 'log2'} or None. Got 'auto' instead.

--------------------------------------------------------------------------------
1 fits failed with the following error:
Traceback (most recent call last):
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selec
tion\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 1144, in wrapper
    estimator._validate_params()
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 637, in _validate_params
    validate_parameter_constraints(
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\utils\_para
m_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter
of RandomForestClassifier must be an int in the range [1, inf), a float in the range
(0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selection\_se
arch.py:976: UserWarning: One or more of the test scores are non-finite: [0.97742129
0.97742129 0.97916042 0.98088456 0.98262369 0.98089955
        nan 0.98088456 0.98088456 0.97742129]
  warnings.warn(
```


Receiver Operating Characteristic (ROC) - Random Forest Classifier and Cross validation, Hyperparameters tuned (Grid)
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
ROC curve (area = 0.93)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate


Random Forest Classifier and Cross validation, Hyperparameters tuned (Grid)
523
3
0
1
True Class
0
1
Predicted Class





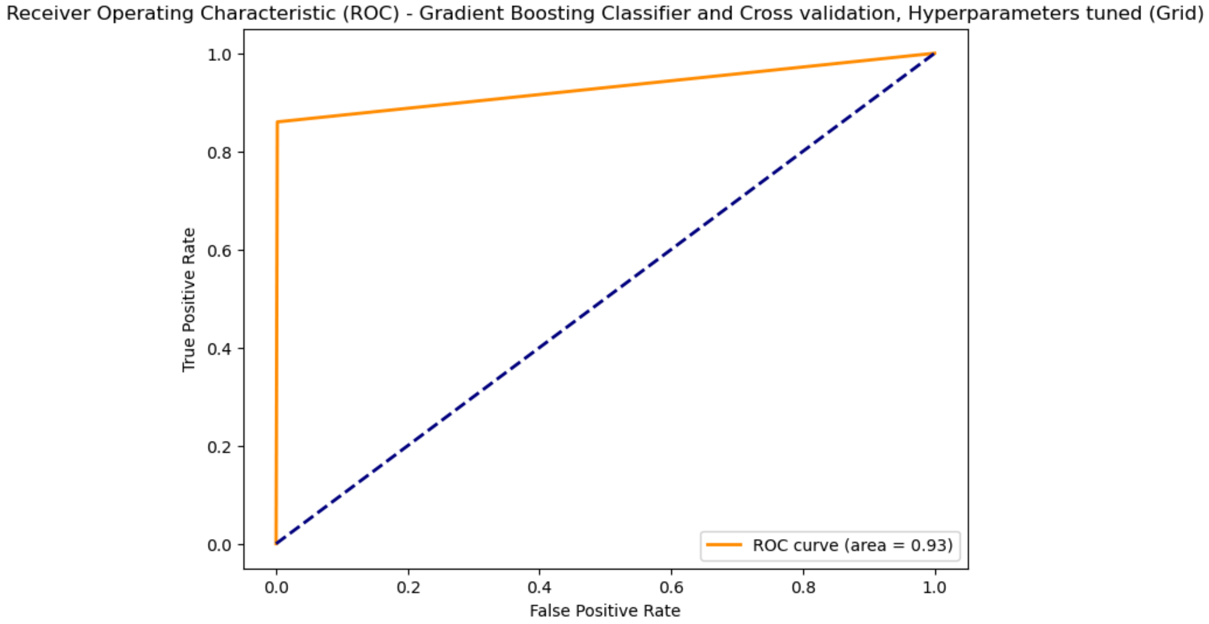
Gradient Boosting Hiperparameters tuning

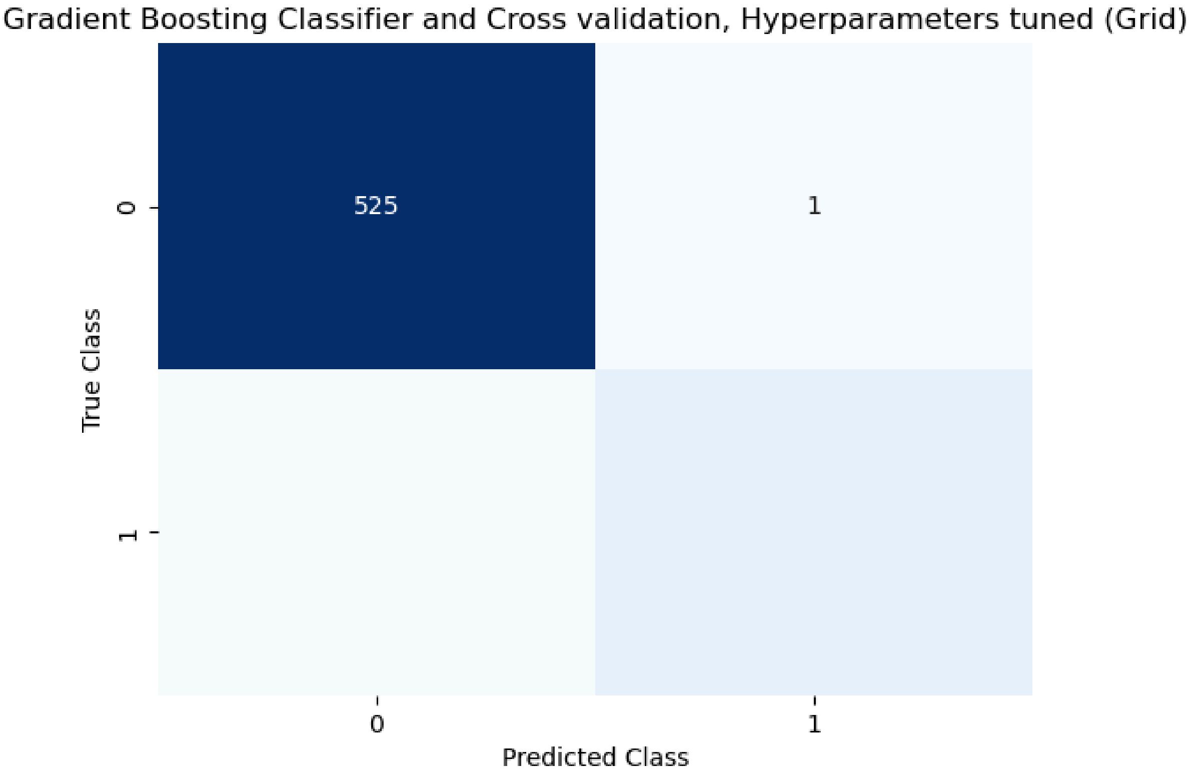
In [ ]:

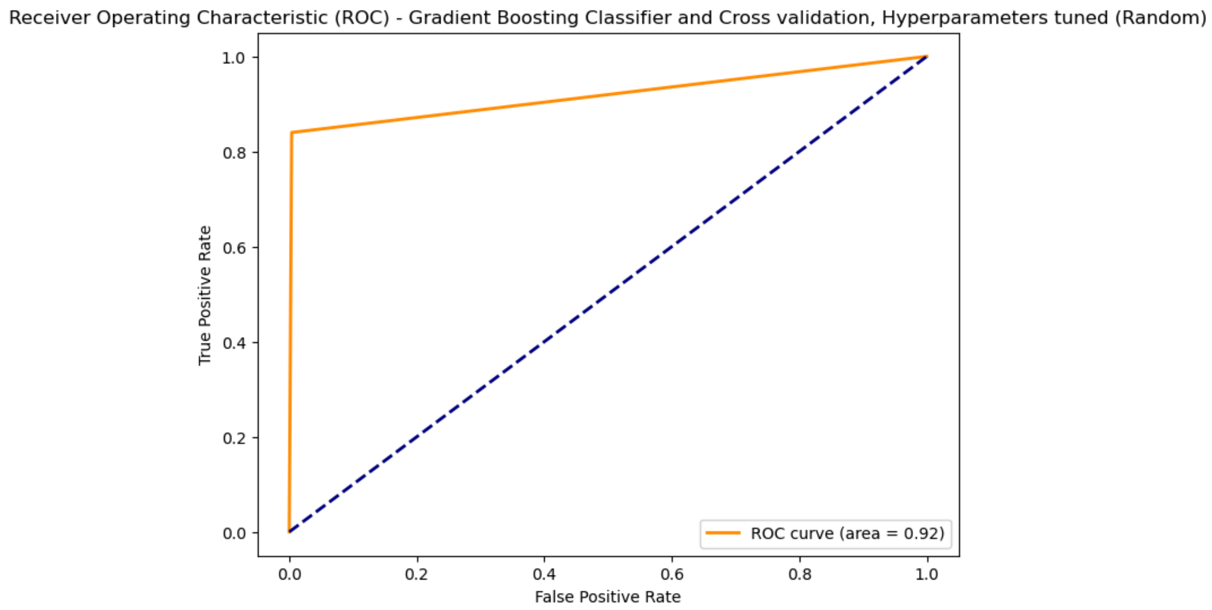
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV

# Create a Random Forest classifier object
gb_classifier = GradientBoostingClassifier()

# Define the hyperparameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'learning_rate': [0.01, 0.1, 0.5],
    'max_depth': [3, 5, 7],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['auto', 'sqrt', 'log2']
}

# Perform randoized search
np.random.seed(42)
random_search = RandomizedSearchCV(estimator=gb_classifier, param_distributions=par
random_search.fit(features, target)

rand_params = random_search.best_params_

## Perform grid search
#grid_search = GridSearchCV(estimator=gb_classifier, param_grid=param_grid, cv=5, n
#grid_search.fit(features, target)
## Best hyperparameters: {'bootstrap': True, 'max_depth': 20, 'max_features': 'log2

## Get the best hyperparameters
#best_params = grid_search.best_params_
#print("Best hyperparameters:", best_params)

best_params = { 'learning_rate': 0.5,
                'max_depth': 7,
                'max_features': 'sqrt',
                'min_samples_leaf': 2,
                'min_samples_split': 10,
                  'n_estimators': 50}

best_params_classifier = GradientBoostingClassifier(**best_params)

model_title = "Gradient Boosting Classifier and Cross validation, Hyperparameters t

gb_cv_hp_grid_results, predictions = myCrossValidation(best_params_classifier, skf,

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

gb_cv_hp_grid_results.head()


rnd_params_classifier = GradientBoostingClassifier(**rand_params)

model_title = "Gradient Boosting Classifier and Cross validation, Hyperparameters t

gb_cv_hp_rnd_results, predictions = myCrossValidation(rnd_params_classifier, skf, f

myPlotROCcurve(target, predictions, model_title)

myPlotConfusionMatrix(target, predictions, model_title)

gb_cv_hp_rnd_results.head()
```

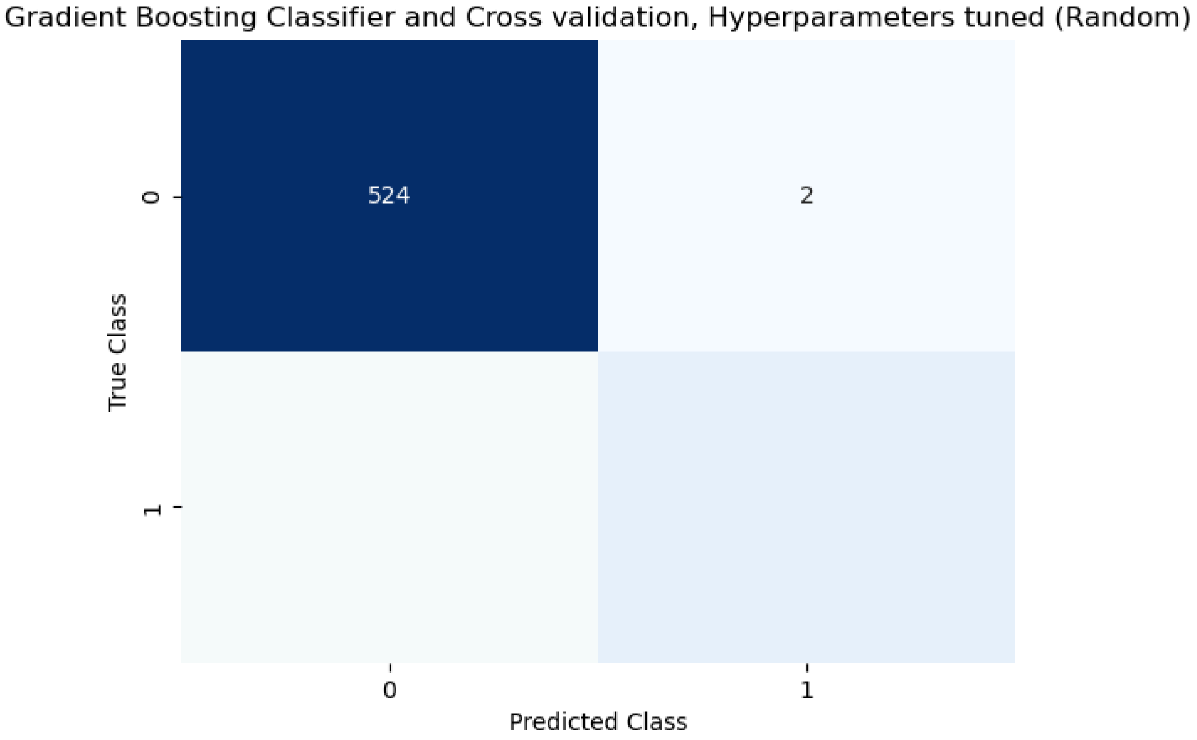
```
final_results.append(gb_cv_hp_grid_results)
final_results.append(gb_cv_hp_rnd_results)
```

```
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selection\_va
lidation.py:425: FitFailedWarning:
15 fits failed out of a total of 50.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score
='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
11 fits failed with the following error:
Traceback (most recent call last):
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selec
tion\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 1144, in wrapper
    estimator._validate_params()
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 637, in _validate_params
    validate_parameter_constraints(
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\utils\_para
m_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter
of GradientBoostingClassifier must be an int in the range [1, inf), a float in the r
ange (0.0, 1.0], a str among {'log2', 'sqrt'} or None. Got 'auto' instead.

--------------------------------------------------------------------------------
4 fits failed with the following error:
Traceback (most recent call last):
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selec
tion\_validation.py", line 732, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 1144, in wrapper
    estimator._validate_params()
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\base.py", l
ine 637, in _validate_params
    validate_parameter_constraints(
  File "c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\utils\_para
m_validation.py", line 95, in validate_parameter_constraints
    raise InvalidParameterError(
sklearn.utils._param_validation.InvalidParameterError: The 'max_features' parameter
of GradientBoostingClassifier must be an int in the range [1, inf), a float in the r
ange (0.0, 1.0], a str among {'sqrt', 'log2'} or None. Got 'auto' instead.

  warnings.warn(some_fits_failed_message, FitFailedWarning)
c:\Users\Nemanja\miniconda3\envs\mlenv\lib\site-packages\sklearn\model_selection\_se
arch.py:976: UserWarning: One or more of the test scores are non-finite: [0.97916042
0.95488756 0.98088456 0.97914543 0.98610195        nan
 0.9131934         nan 0.95487256        nan]
  warnings.warn(
```


Receiver Operating Characteristic (ROC) - Gradient Boosting Classifier and Cross validation, Hyperparameters tuned (Grid)
True Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.93)

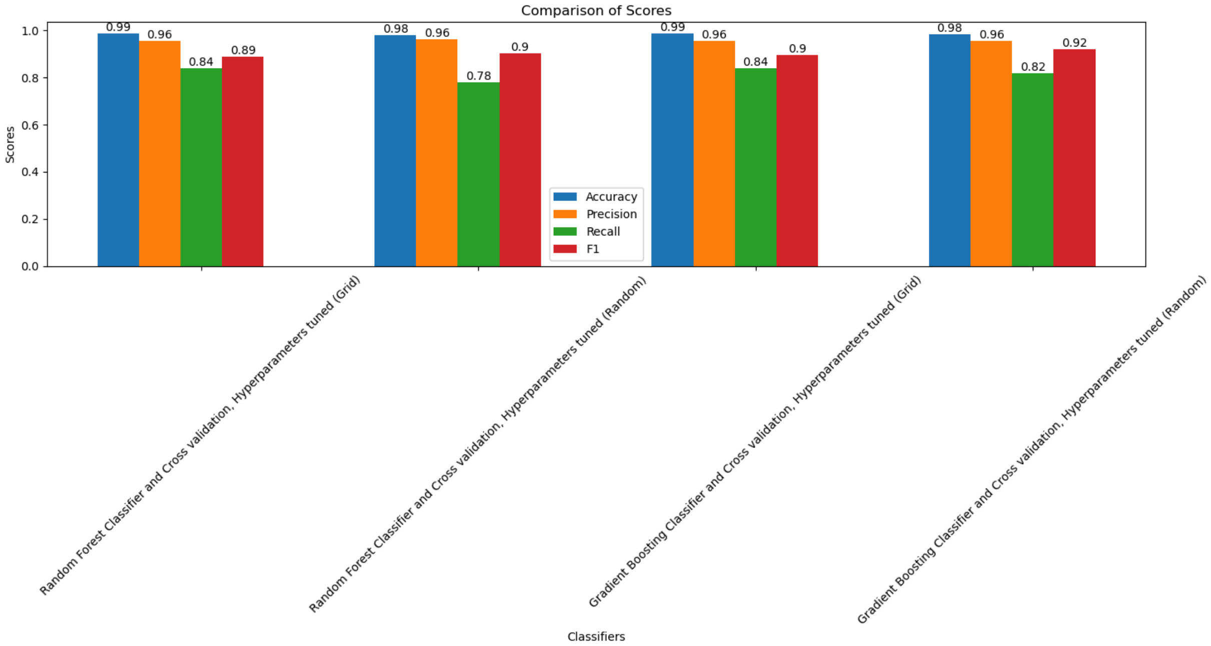

Gradient Boosting Classifier and Cross validation, Hyperparameters tuned (Grid)
True Class
0
1
525
1
Predicted Class
0
1





Comparison

```
In [ ]: plot_classifier_scores(final_results, feature_names, angle = 45)
```



Zaključujemo da je najbolje rezultate dao Gradient Boosting algoritam sa isprobavanjem svih varijacija parametara, dok je random biranje parametara dalo slične rezultate za oba algoritma.