

Introduction

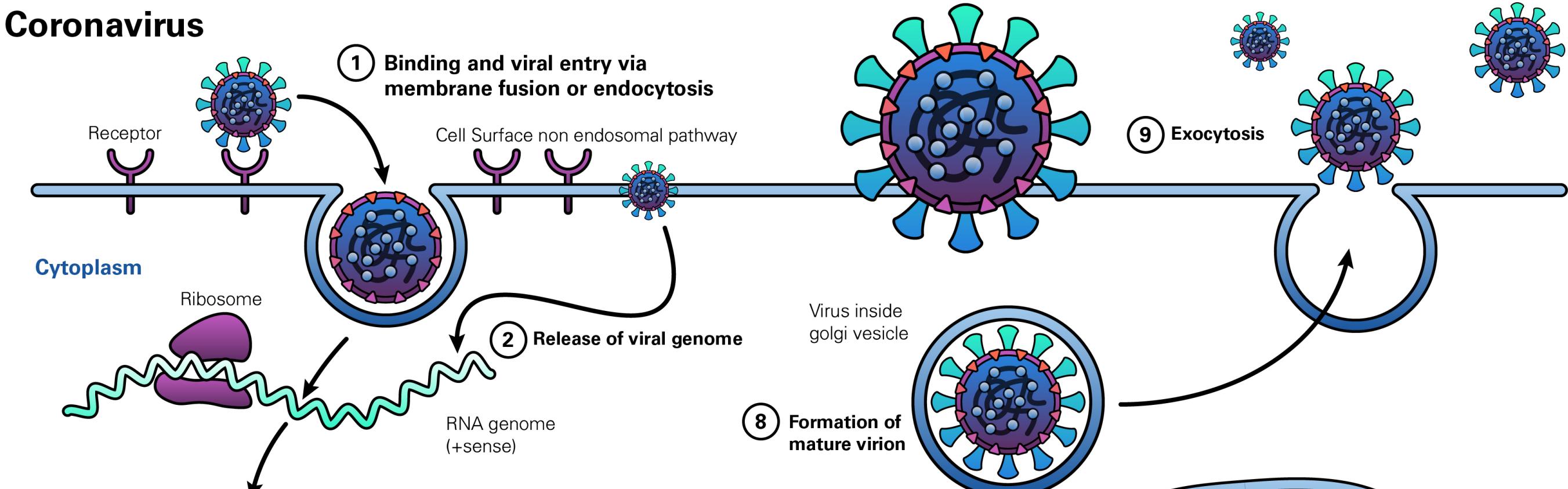
In this report, the biology of sars-cov-2 is discussed and the sars-cov-2 genome is analyzed. Machine learning will be used predict the translation transformations and deep learning will be used to predict the next nucleotide in the genome sequence.

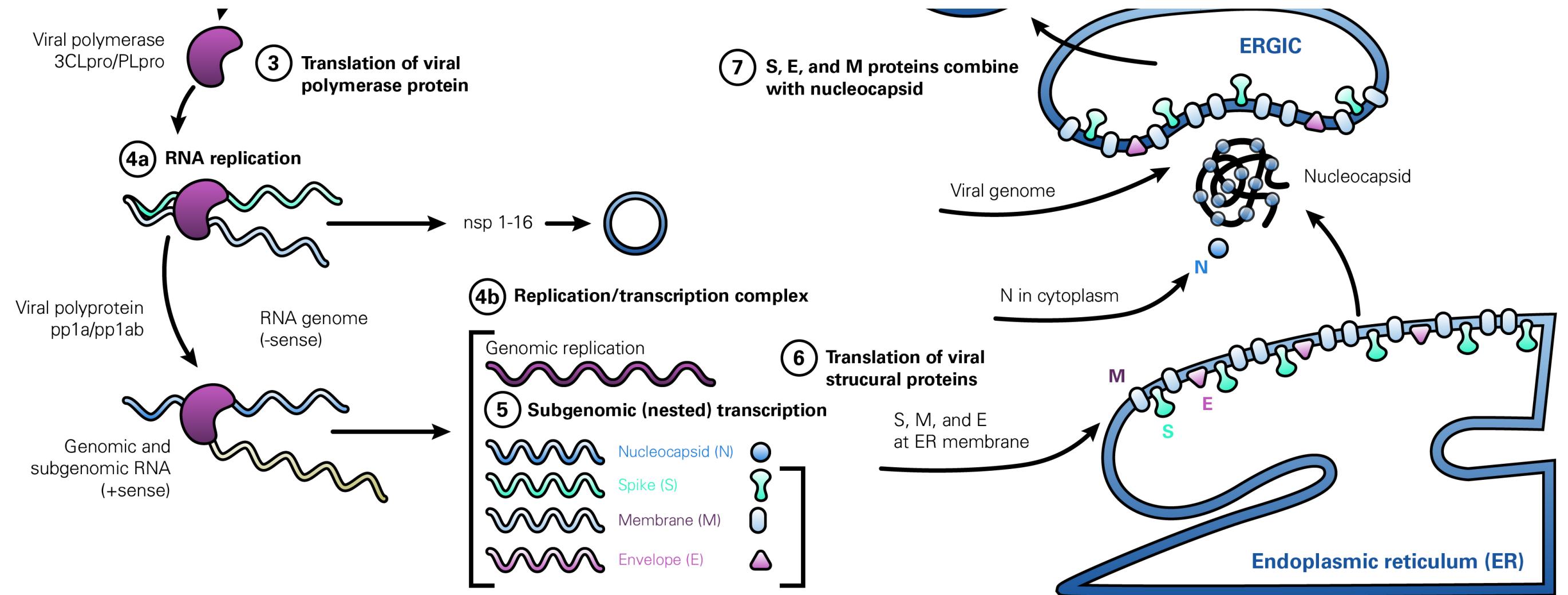
SARS-CoV-2 Biology

Coronaviruses are a sphere-like 20-sided polygon, icosahedron, shape and belong to the Nidovirales order. SARS-CoV-2 travels through the respiratory tract and attaches itself to alveoli, air sacs, in the lungs. Coronaviruses are named after the dyed red spike glycoproteins (S) that cover them and attach these proteins by being activated by the host cell's proteases, ACE2 and TMPRSS2 enzymes, at a sequential proteolytic cleavage S1 and S2 site to fuse with a host cell's enzymes to enter, endocytosis, the cell's plasma membrane to the cytosol. S1 is the end binding region of the S protein and S2 is the beam of the S protein. The spike glycoproteins can also attach to a cell's endosomes being activated by the host cell's CTSL enzyme. Inside of the icosahedron positive-sensed lipid bilayer envelope is a helical capsid that encloses an RNA genome (gRNA), which encodes the viral genetic information -- non-segmented-single-stranded RNA -- that is used during the lytic cycle to generate RNA polymerase, protease, and other proteins. The envelope is removed allowing genomic RNA to enter the cytoplasm where some of it is translated into pp1a and pp1ab proteins. Translation is when codons, groups of three nucleotides, move along a ribosome through tRNA that translates the codons to anticodons that generate an amino acid polypeptide chain of proteins. Both proteins are then cleaved by protease making 16 nonstructural proteins, NSPs. Some of the NSPs form a replication and transcription complex (RTC) in which gRNA is replicated and RNA-dependent RNA polymerase (RdRp) uses gRNA as a template to transcribe (-) subgenomic RNA (sgRNA) and then transcribe it back to (+) sgRNA, which is used as the genome of the new virus. sgRNA is translated into structural proteins, which are spike protein (S), envelope protein (E), membrane protein (M), and nucleocapsid protein (N). Nucleocapsid proteins assemble the viral replication-transcription complexes (RTCs), where gRNA and sgRNAs are synthesized. Spike, envelope, and membrane proteins enter the endoplasmic reticulum, and the nucleocapsid protein encloses the sgRNA becoming a nucleoprotein complex. The proteins and nucleoprotein complex combine into a virus endoplasmic reticulum-Golgi apparatus, and then are excreted into a vesicle. The dyed yellow envelope proteins (E) and dyed orange membrane glycoproteins (M) assist in budding -- using the host cell's plasma membrane for envelope formation -- and then exiting, exocytosis, the host cell through lysis. The following is a diagram of the SARS-CoV-2 lifecycle,

In [4]: `Image(filename='Coronavirus_Life_Cycle.jpg')`

Out[4]:

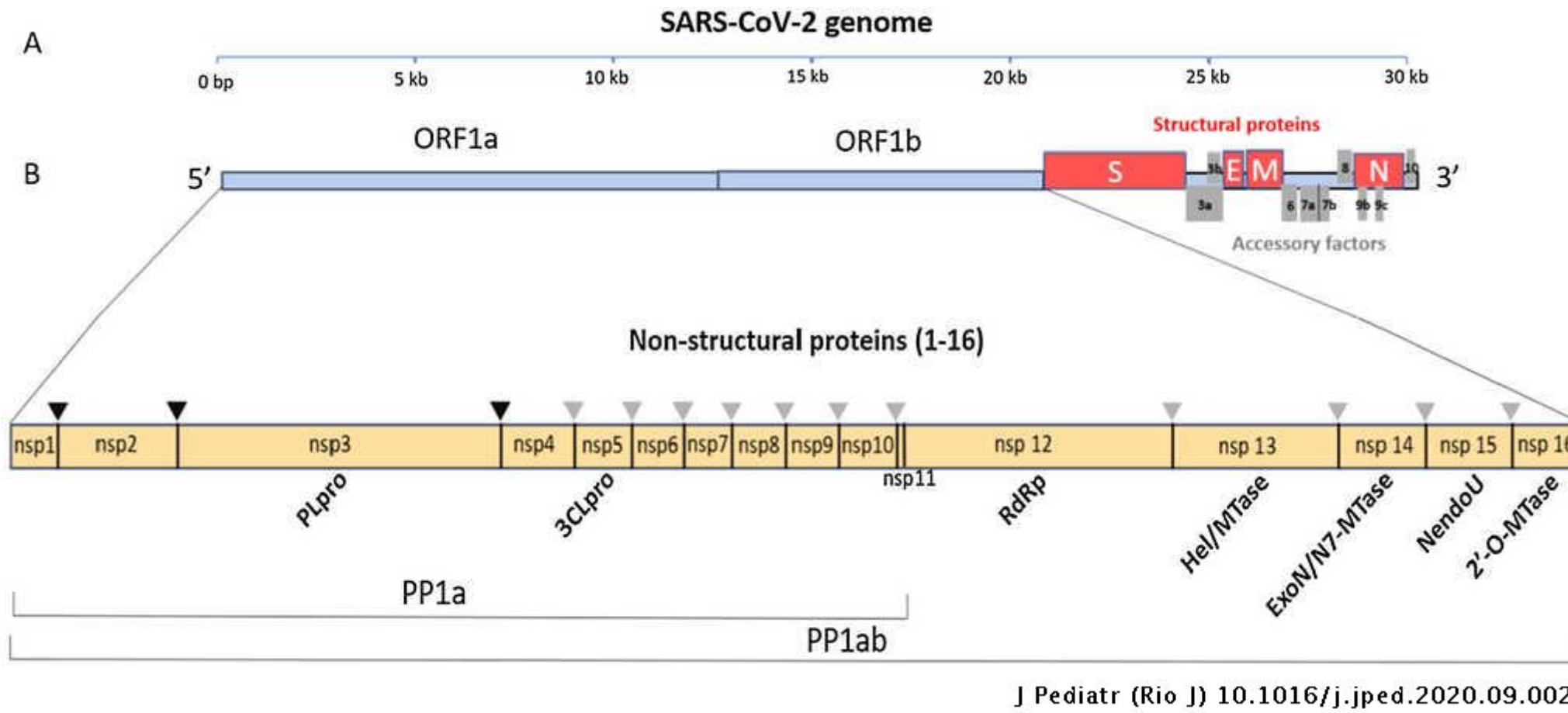




Sars-cov-2 has the largest RNA viral genome, which ranges from 26 to 32 kb. The genome of the virus starts with 5' carbon base, has an ORF1ab, S, E, M, N, and accessory region, and then end with 3' carbon base. ORF1ab are non-structural proteins. ORF1ab, open reading frames, comprises of 16 NSP regions. Nsp1 and nsp2 encode enzymes that bind to RNA and suppress gene expression. Nsp3 encodes enzymes that assist in the translation of viral mRNAs. Nsp4 encodes viroporin, which modify cellular membranes. Nsp5 encodes the organization of the viroplasm, where viral replication and assembly occur. Nsp6 generates autophagosomes, vesicles with cellular material that will be removed through autophagy. Nsp7 through 16 encode the generation of RNA synthesis and processing. Nsp7 through 11 encode the contribution to the nsp interactome, molecular interactions. Nsp12 encodes RNA polymerase. Nsp13 encodes helicase. Nsp14 through 16 encode mRNA capping, attaches the 5' cap to mRNA. The S region encodes the spike proteins, which bind to host cell receptors and fuse to the cell membrane. The M and E regions encode the membrane and envelope proteins, which assist in budding. The N region encodes the nucleocapsid protein, which assembles the viral replication-transcription complexes. The genome additionally contains an ORF coding for accessory proteins, which are not essential in virus replication but assist in pathogenesis, development of the virus.

```
In [13]: Image(filename='SARS-COV-2_Genome.jpeg')
```

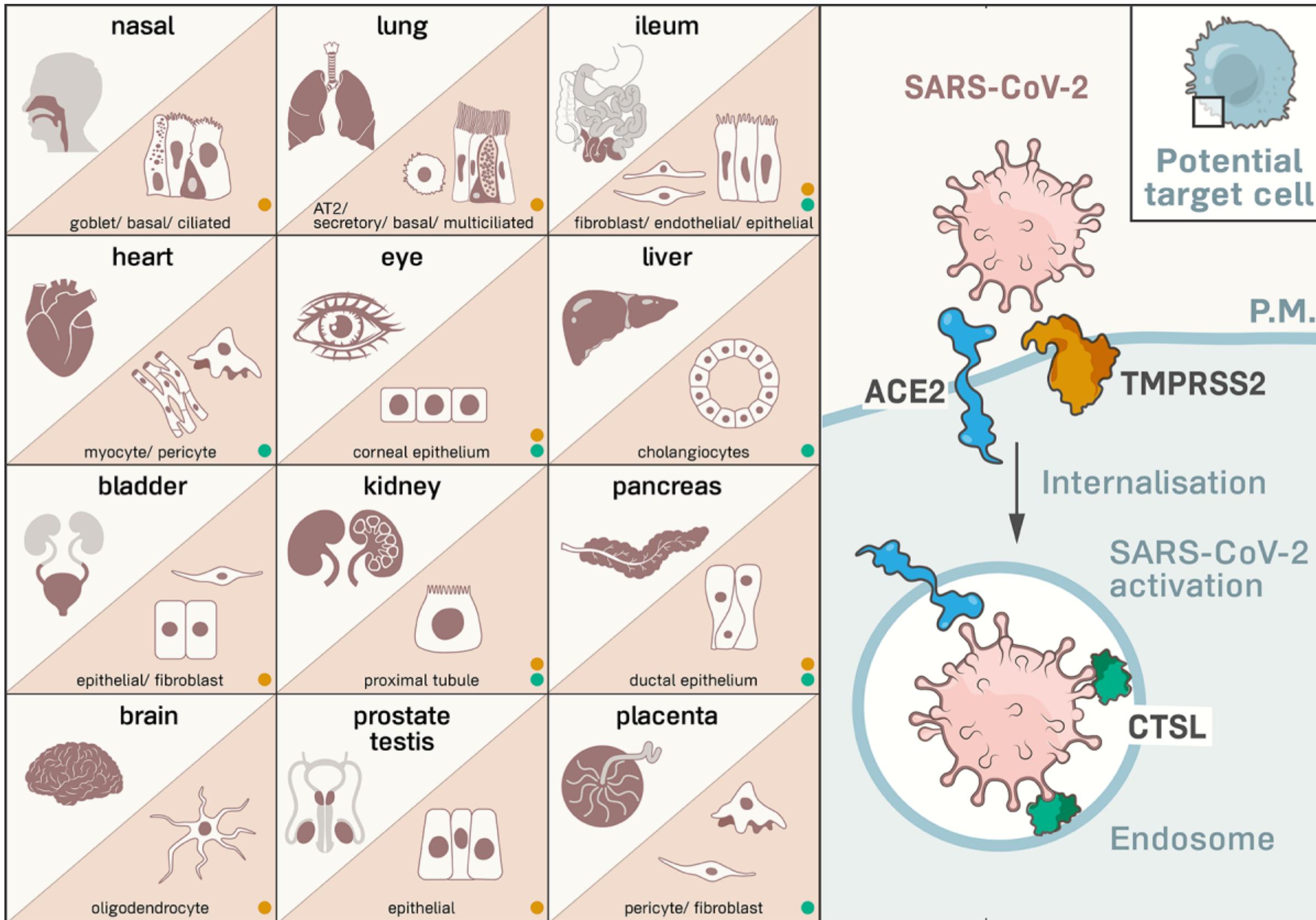
```
Out[13]:
```



ACE2 modulates the activity of the angiotensin II protein, which increases blood pressure and inflammation. The SARS-CoV-2 virus binding to ACE2 inhibits ACE2 and can cause an increase in cell inflammation resulting in the death of cells in the alveoli units with hypoxic fluid flooding, an increase in blood pressure damaging blood vessels causing microvascular thrombosis, and the transduction of the kidney podocytes resulting in acute kidney injury. The following is a diagram of the enzymes that get inhibited and the parts of the body that can get damaged.

```
In [5]: Image(filename='ace2-thumb-1.png')
```

```
Out[5]:
```



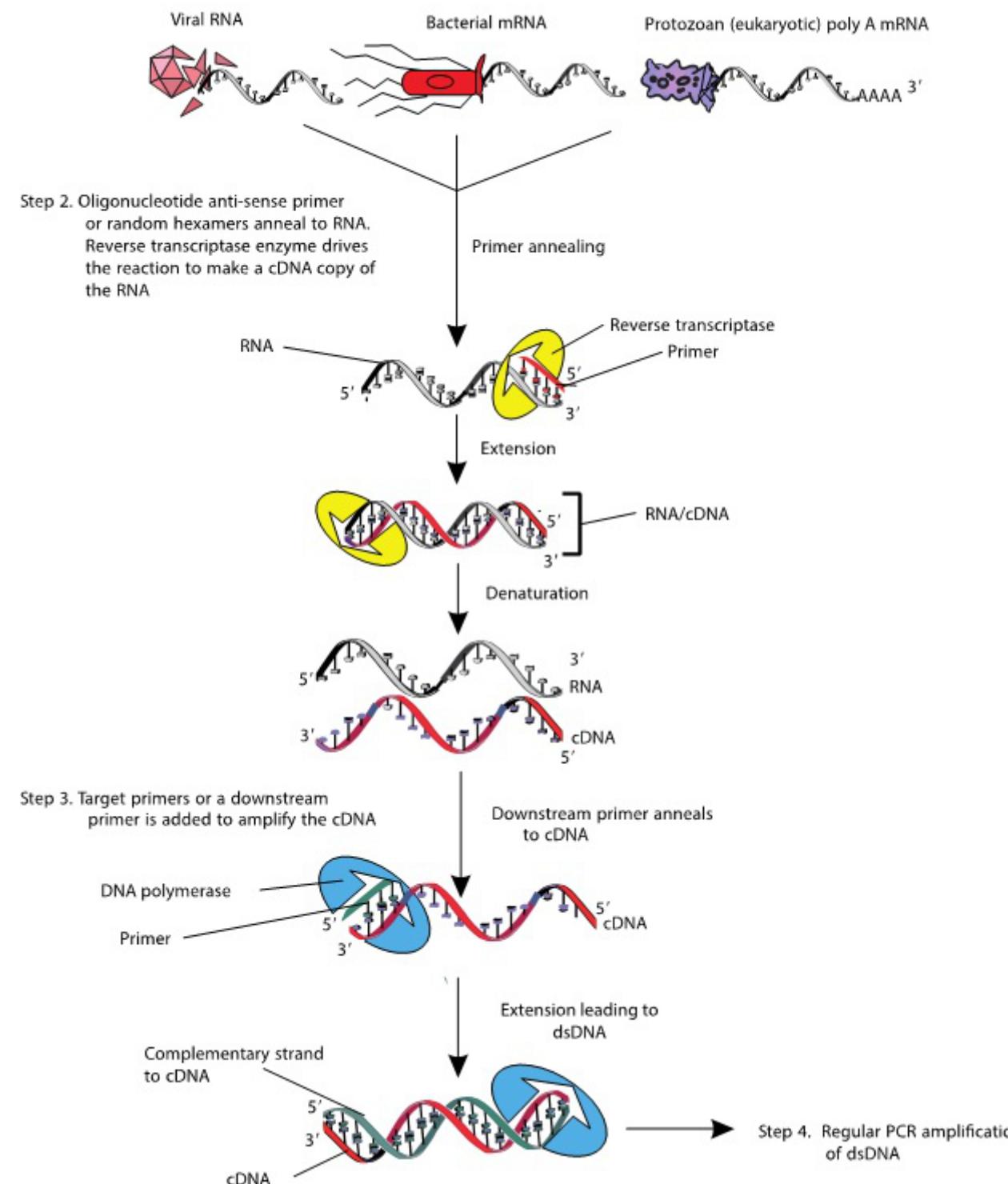
Testing for SARS-CoV-2

A patient can be tested for SARS-CoV-2 with the sequence-specific molecular nucleic acid assay, or the antigen-specific immunoassay. For a molecular-assay, a nasal or saliva sample is collected from upper respiratory fluid and then Real-time RT-PCR is conducted to quantify sequences within the RNA samples. Reverse transcriptase converts extracted RNA into cDNA that is used as a template for DNA polymerase to complete the strand of dsDNA and then RT-PCR amplifies the genetic regions and fluorescent probes bind to the regions for identification. For an immuno-assay, antibodies are put on a membrane and complexed with a potentially virulent

sample of which any antigen will be trapped that results in the membrane changing color. A SARS-CoV-2 recovered patient can be tested for whether the patient has developed antibodies against the virus by checking a blood sample for the antibodies. The following are diagrams of a molecular assay

In [6]: `Image(filename='10104fig1.jpg')`

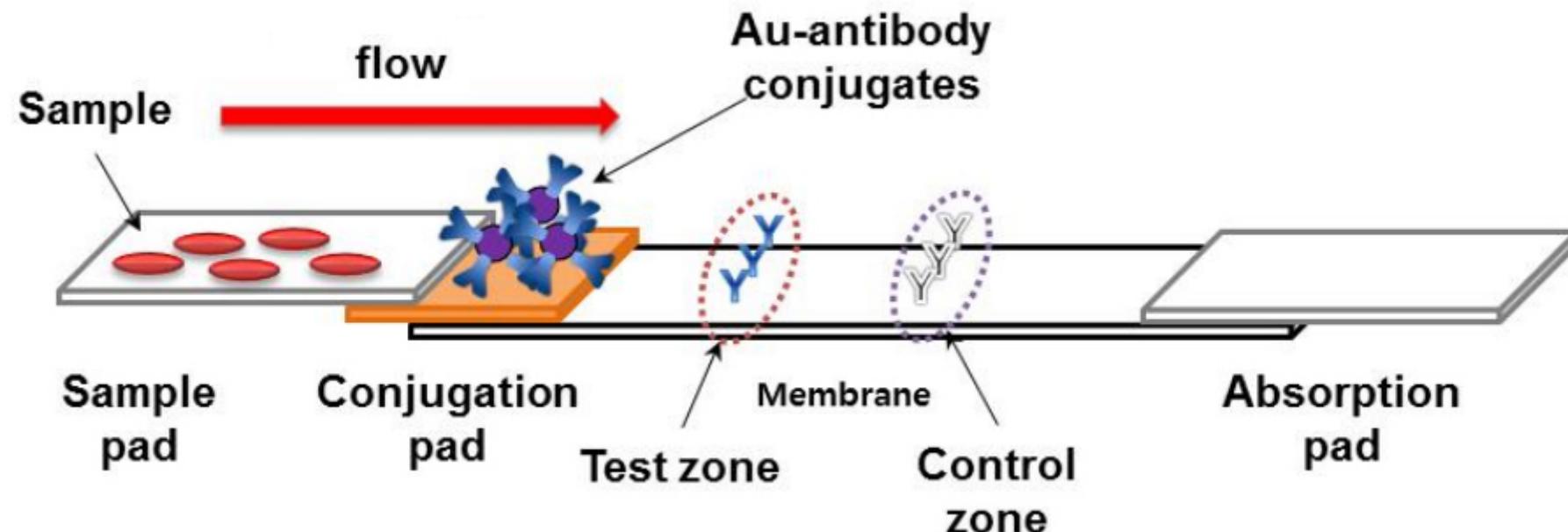
Out[6]: Step 1. Target RNA is isolated from the sample



and an immuno assay.

In [7]: `Image(filename='tag-lateral-flow-immunoassay-2.jpg')`

Out[7]:

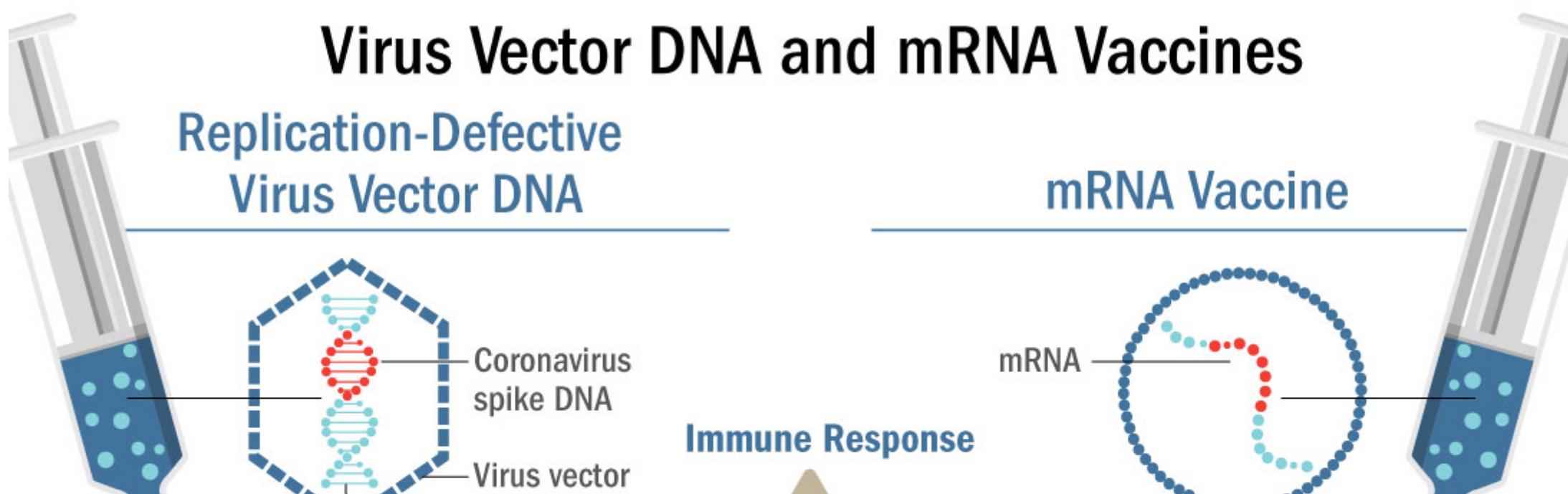


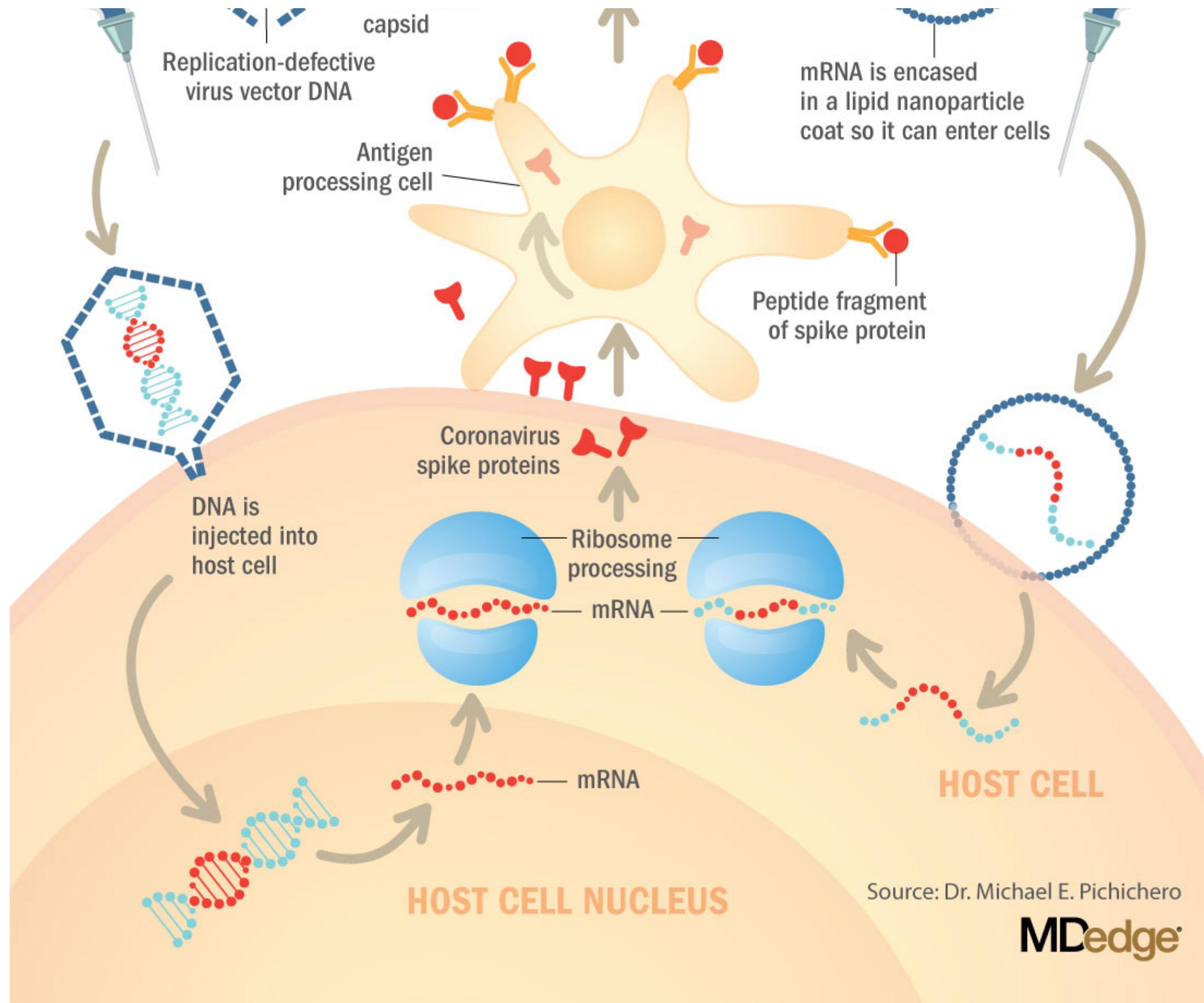
Vaccines for SARS-CoV-2

Vaccines have been developed to combat SARS-CoV-2. The Pfizer/BioNTech vaccine (BNT162b2) and The ModernaTX, Inc vaccine (mRNA-1273) are mRNA vaccines. mRNA vaccines deliver lab designed viral mRNA made by coding both the 5'-untranslated region (UTR) and the 3'-UTR into cells where it is translated into the encoded antigen to which an immune recognition by white blood cells, immunogenicity, results. The AstraZeneca in collaboration with the University of Oxford vaccine (AZD1222) is an adenovirus vaccine. Adenovirus vaccines are first gene sub-cloned into an intermediary vector, transferred to an adenovirus recombinant genome vector, transfected into packaging cells, amplified into a culture stock, and then titrated to determine the concentration of active adenoviruses in the stock. The adenoviruse is then delivered to cells where immunogenicity results. The following is a diagram of Virus vector DNA and mRNA vaccine immune response production. Vaccines have been developed to combat SARS-CoV-2. The Pfizer/BioNTech vaccine (BNT162b2) and The ModernaTX, Inc vaccine (mRNA-1273) are mRNA vaccines. mRNA vaccines deliver lab designed viral mRNA made by coding both the 5'-untranslated region (UTR) and the 3'-UTR into cells where it is translated into the encoded antigen to which an immune recognition by white blood cells, immunogenicity, results. The AstraZeneca in collaboration with the University of Oxford vaccine (AZD1222) is an adenovirus vaccine. Adenovirus vaccines are first gene sub-cloned into an intermediary vector, transferred to an adenovirus recombinant genome vector, transfected into packaging cells, amplified into a culture stock, and then titrated to determine the concentration of active adenoviruses in the stock. The adenoviruse is then delivered to cells where immunogenicity results. The following is a diagram of Virus vector DNA and mRNA vaccine immune response production.

```
In [8]: Image(filename='DNA_and_mrNA_Vaccines_web.jpeg')
```

```
Out[8]:
```





Import data of sars-cov-2 genome.

```
df=Json('sars-cov-2_genome')
```

Source: Dr. Michael E. Pichichero

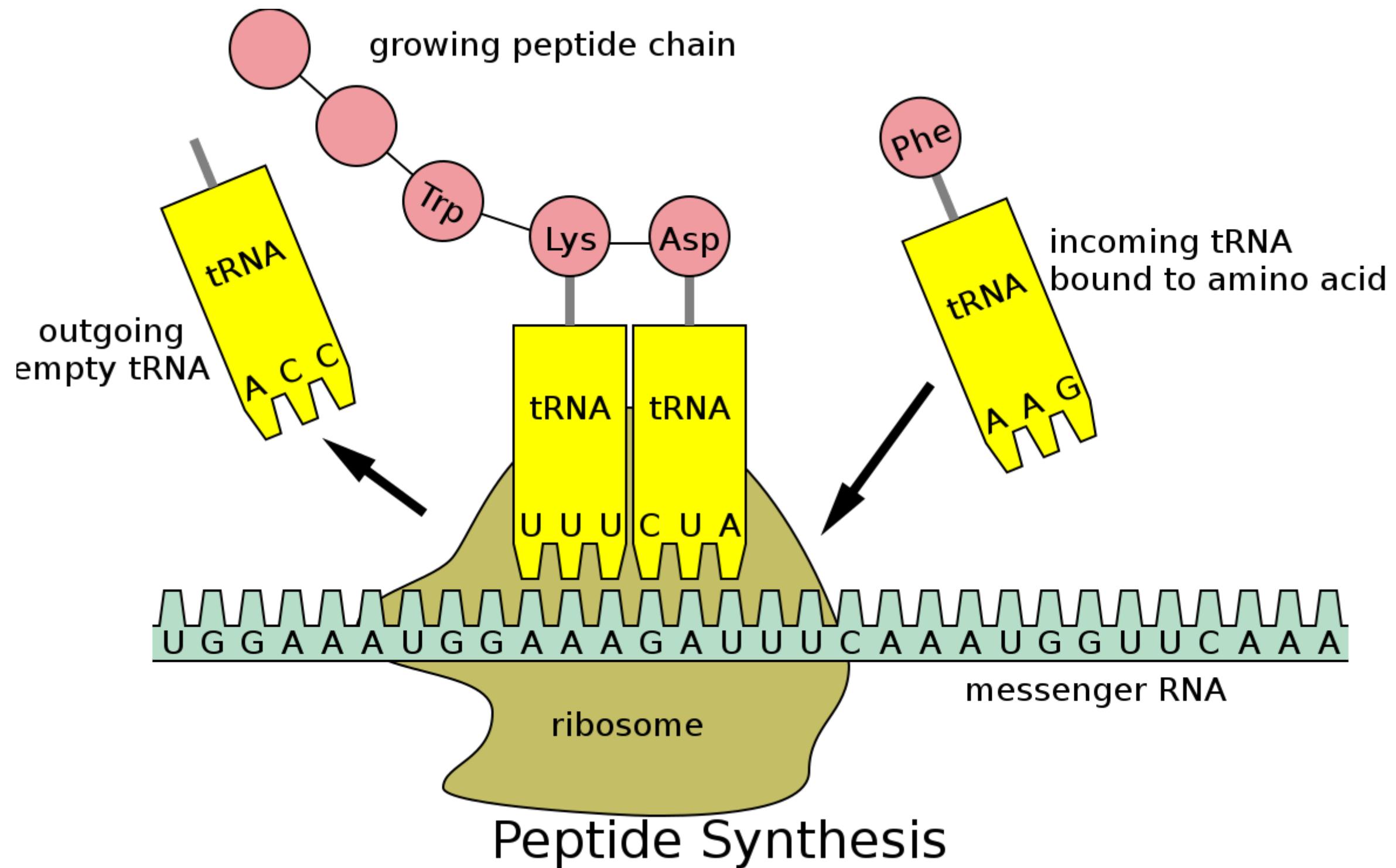
MDedge®

```
In [296...]: df.file("genome.fna")
genome=json_storage['sars-cov-2_genome']
genome
```


Translation has three steps: initiation, which is when a ribosome, mRNA (sgRNA is a type of mRNA), and tRNA connect; elongation, which is when tRNA brings amino acids to the ribosome to be linked; and termination, which is when the polypeptide chain is complete. In this report, machine learning is used in two steps to complete the elongation process. First, the algorithm will bind the codons to anticodons and then it will transfer the anticodons to amino acids.

```
In [225]: Image(filename='translation.png')
```

Out[225...]



Data Mining

The genome a '\n' character that is python for new line that will be removed.

In [192... set(genome)

Out[192... {'\n', 'A', 'C', 'G', 'U'}

Data Cleaning

Remove '\n' from genome and protein strings.

```
In [297...]: genome=genome.replace('\n','')
          genome
```


Makes a dataframe of sars-cov-2 mRNA nucleotides and mRNA antinucleotide

```
In [298...]: translation_df1=pd.DataFrame(columns=[ 'mRNA_Codons' , 'mRNA_Codons_copy' , 'Anticodons' ])
for i in genome:
    translation_df1.loc[len(translation_df1.index)] = [ i , i , i ]
translation_df1
```

Out[298...]	mRNA_Codons	mRNA_Codons_copy	Anticodon
0	A		A
1	C		C
2	U		U
3	U		U
4	U		U
...
27464	U		U
27465	G		G
27466	A		A
27467	U		U
27468	A		A

27469 rows × 3 columns

Translate codons into corresponding anticodons

```
In [299]: translation_df1['Anticodons'] = translation_df1['Anticodons'].replace('A', 'T')
translation_df1['Anticodons'] = translation_df1['Anticodons'].replace('U', 'A')
translation_df1['Anticodons'] = translation_df1['Anticodons'].replace('T', 'A')
translation_df1['Anticodons'] = translation_df1['Anticodons'].replace('C', 'G')
```

```
translation_df1['Anticodons']=translation_df1['Anticodons'].replace('G','C')
translation_df1['Anticodons']=translation_df1['Anticodons'].replace('2','G')
translation_df1
```

Out[299...]

	mRNA_Codons	mRNA_Codons_copy	Anticodons
0	A	A	U
1	C	C	G
2	U	U	A
3	U	U	A
4	U	U	A
...
27464	U	U	A
27465	G	G	C
27466	A	A	U
27467	U	U	A
27468	A	A	U

27469 rows × 3 columns

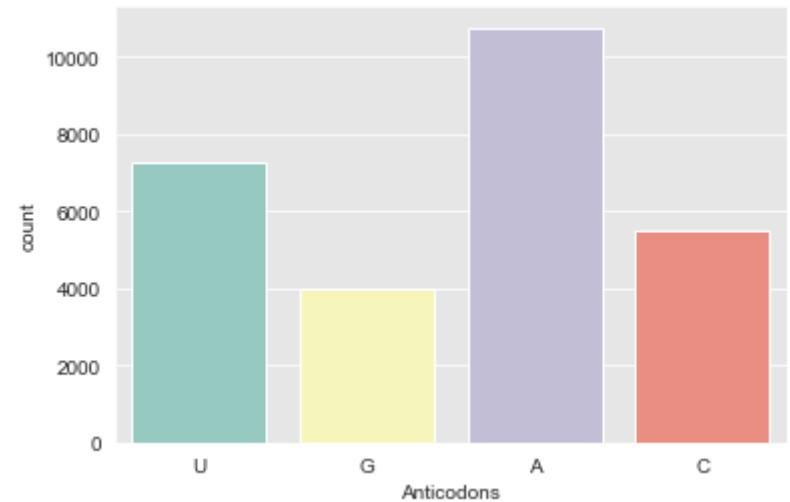
Data Exploration

Sars-cov-2 genome is made up of apoximativly 30% adenine, 18% guanine, 20% cytosine and 32% uracil.

In [199...]

```
sns.countplot(translation_df1['Anticodons'], palette='Set3')
```

Out[199...]



Percentage of uracil in genome data.

In [200...]

```
len(translation_df1['Anticodons'].loc[translation_df1['Anticodons']=='U'])/len(translation_df1['Anticodons'])*100
```

Out[200...]

26.338781899595908

Percentage of guanine in genome data.

```
In [201]: len(translation_df1['Anticodons'].loc[translation_df1['Anticodons']=='G'])/len(translation_df1['Anticodons'])*100  
Out[201]: 14.49998179766282
```

Percentage of adenine in genome data.

```
In [202]: len(translation_df1['Anticodons'].loc[translation_df1['Anticodons']=='A'])/len(translation_df1['Anticodons'])*100  
Out[202]: 39.14594633950999
```

Percentage of cytosine in genome data.

```
In [203]: len(translation_df1['Anticodons'].loc[translation_df1['Anticodons']=='C'])/len(translation_df1['Anticodons'])*100  
Out[203]: 20.01528996323128
```

There is a class imbalance that will be resolved by oversampling.

Feature Engineering

Represent nucleotides with numbers.

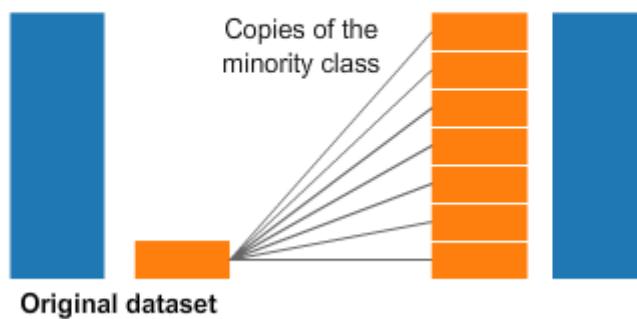
```
In [300]: translation_df1['mRNA_Codons']=translation_df1['mRNA_Codons'].astype('category')  
translation_df1['Anticodons']=translation_df1['Anticodons'].astype('category')  
translation_df1['mRNA_Codons']=translation_df1['mRNA_Codons'].cat.codes  
translation_df1['Anticodons']=translation_df1['Anticodons'].cat.codes
```

Oversampling selects data from minority class population to make up a larger share of the sample.

```
In [9]: Image(filename='oversampling.png')
```

Out[9]:

Oversampling



```
In [301]: oversample = RandomOverSampler(sampling_strategy='minority')
```

Reshape X and y so that they are 2d.

```
In [302]: X1=translation_df1['mRNA_Codons'].values.reshape(-1, 1)  
y1=translation_df1['Anticodons'].values.reshape(-1, 1)
```

Resample independent and dependent variable.

```
In [305...]: x1_resampled, y1_resampled = oversample.fit_resample(x1, y1)
```

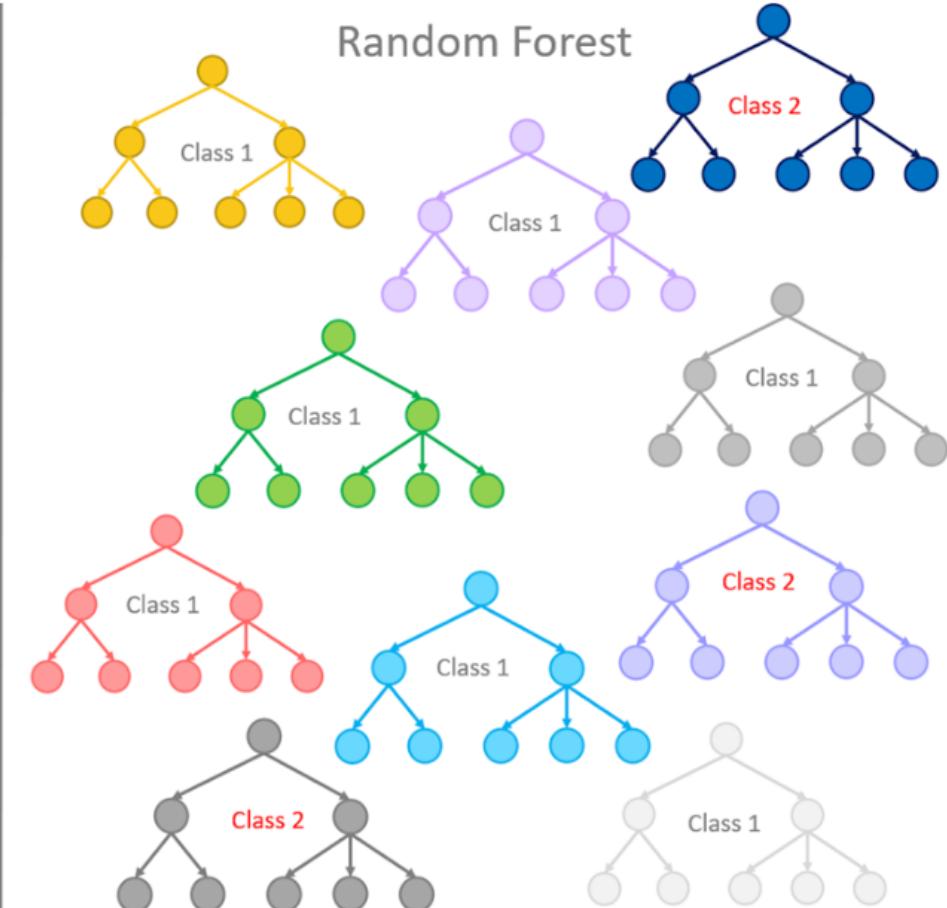
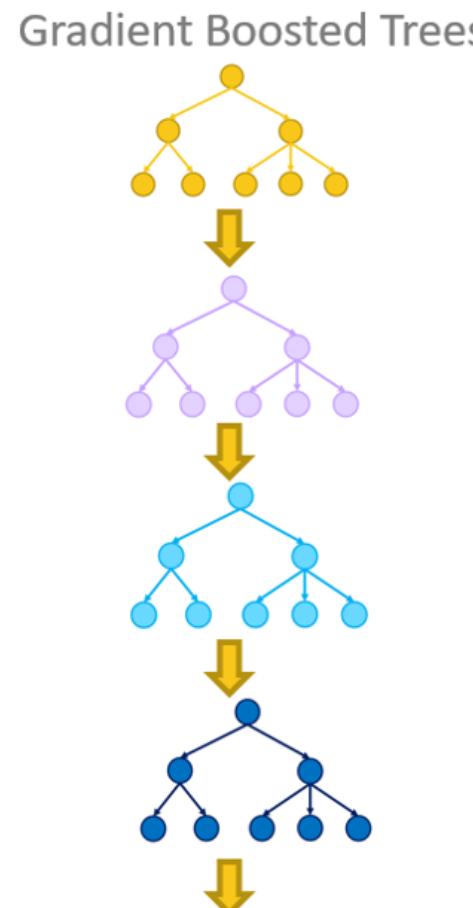
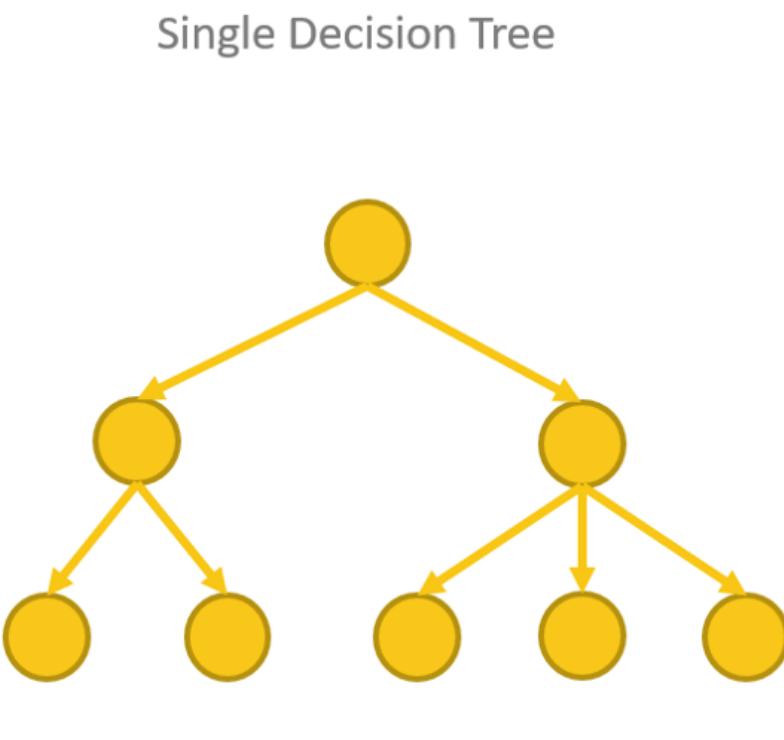
Make a train test split of 80/20.

```
In [306]: x1_train, x1_test, y1_train, y1_test = train_test_split(X1_resampled, y1_resampled, test_size=0.20, random_state=10)
```

Machine Learning Modeling

The following models will be used.

```
In [224]: Image(filename='trees.png')
```



Decision Trees split a node into two or more sub-nodes until node impurity is reduced and information gain is maximized.

Fit the model to the training data.

```
In [307]: decision_tree = DecisionTreeClassifier()  
decision_tree.fit(X1_train, y1_train)
```

```
Out[307]: DecisionTreeClassifier()
```

Show hyperparameters.

```
In [143]: decision_tree.get_params()
```

```
Out[143]: {'ccp_alpha': 0.0,
 'class_weight': None,
 'criterion': 'gini',
 'max_depth': None,
 'max_features': None,
 'max_leaf_nodes': None,
 'min_impurity_decrease': 0.0,
 'min_impurity_split': None,
 'min_samples_leaf': 1,
 'min_samples_split': 2,
 'min_weight_fraction_leaf': 0.0,
 'presort': 'deprecated',
 'random_state': None,
 'splitter': 'best'}
```

The algorithm uses the gini index to reduce impurity during desicion splits. c is the number of classes and p is the probability of the class.

$$Gini = 1 - \sum_i^c (p_i)^2$$

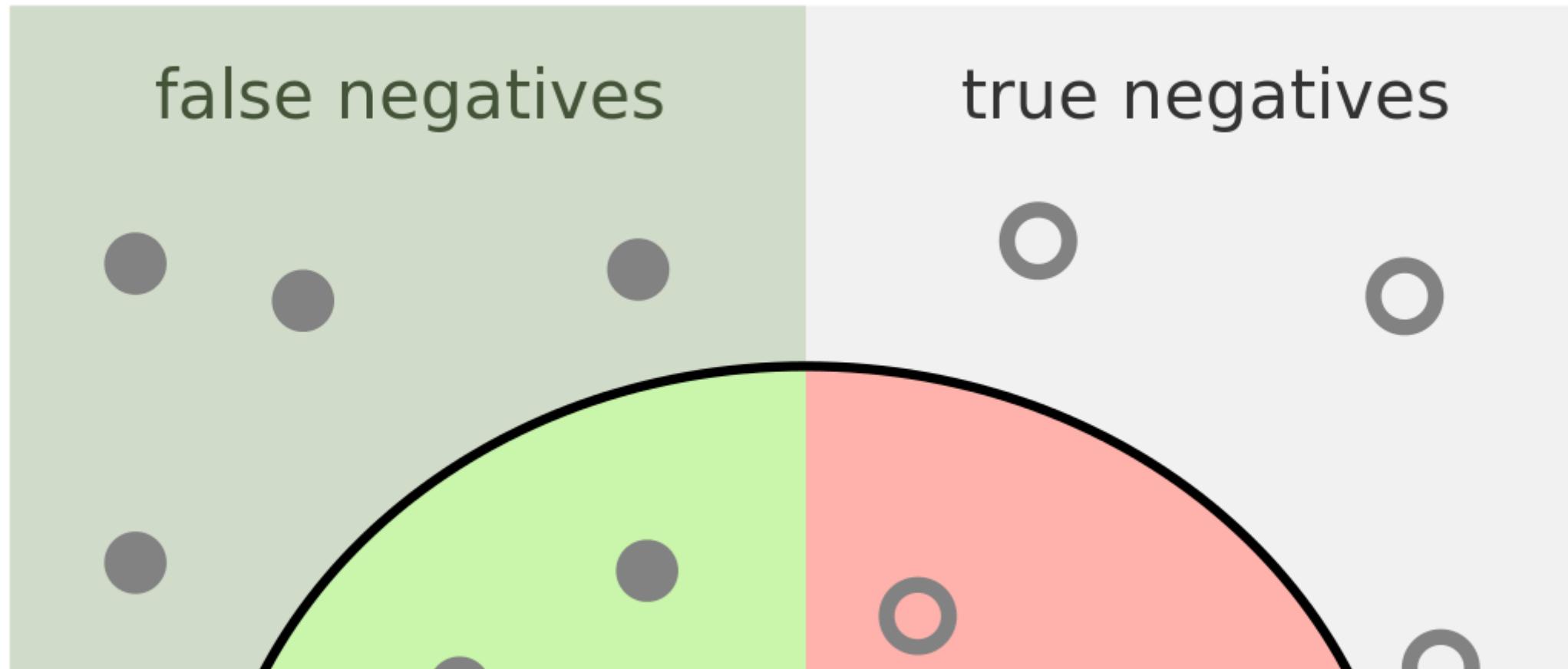
Make predictions.

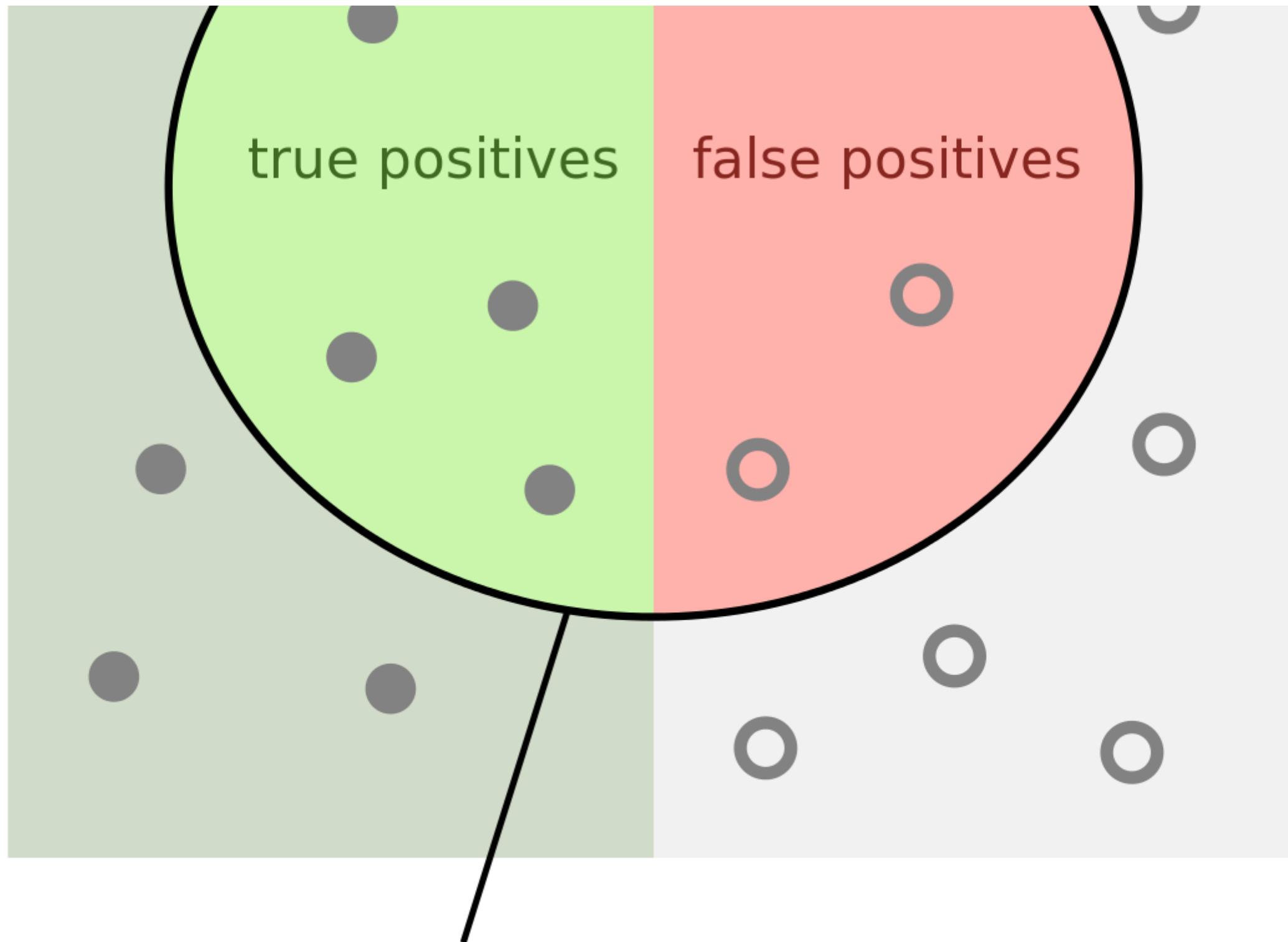
```
In [90]: y1_hat_train_dt = decision_tree.predict(x1_train)
```

```
In [325]: Image(filename='tptnfpfn.png')
```

```
Out[325]:
```

relevant elements



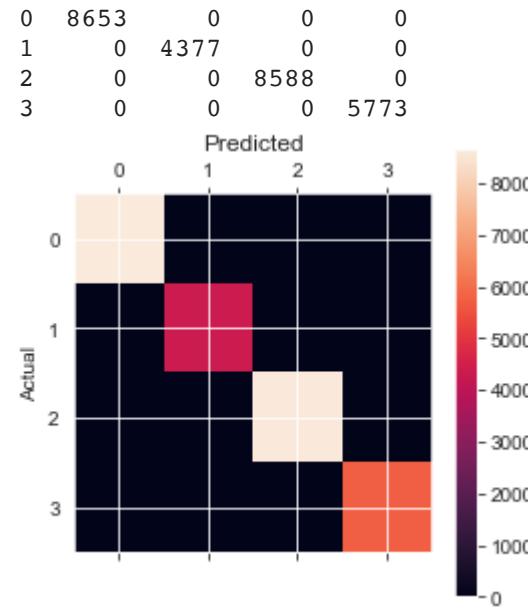


All predictions were classified correctly.

In [91]: `con_mat(y1_train,y1_hat_train_dt)`

Confusion Matrix

0 1 2 3



Separate actual and predicted values into each class for metrics.

```
In [92]: actual_class0=y1_train==0
actual_class1=y1_train==1
actual_class2=y1_train==2
actual_class3=y1_train==3
predictions_class0=y1_hat_train_dt==0
predictions_class1=y1_hat_train_dt==1
predictions_class2=y1_hat_train_dt==2
predictions_class3=y1_hat_train_dt==3
a=[actual_class0,actual_class1,actual_class2,actual_class3]
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3]
```

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$f1 = \frac{2 * precision * recall}{precision + recall}$$

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

$$specificity = \frac{TN}{TN + FP}$$

$$TruePositiveRate = \frac{TP}{TP + FN}$$

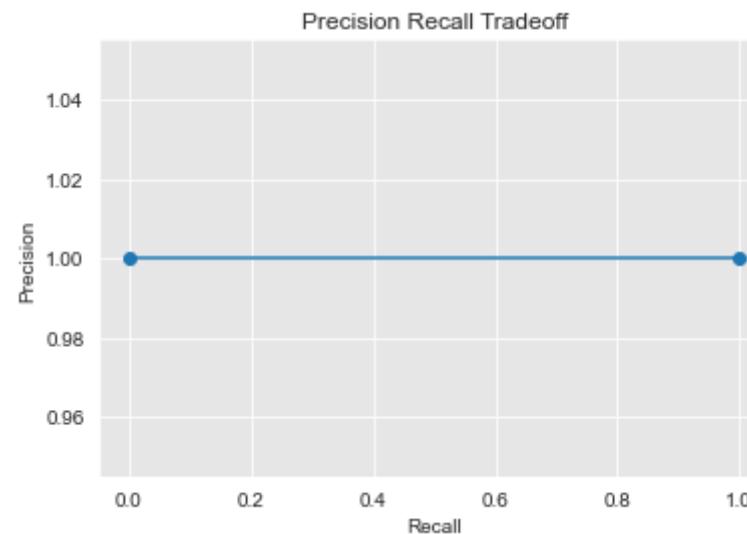
$$FalsePositiveRate = \frac{FP}{FP + TN}$$

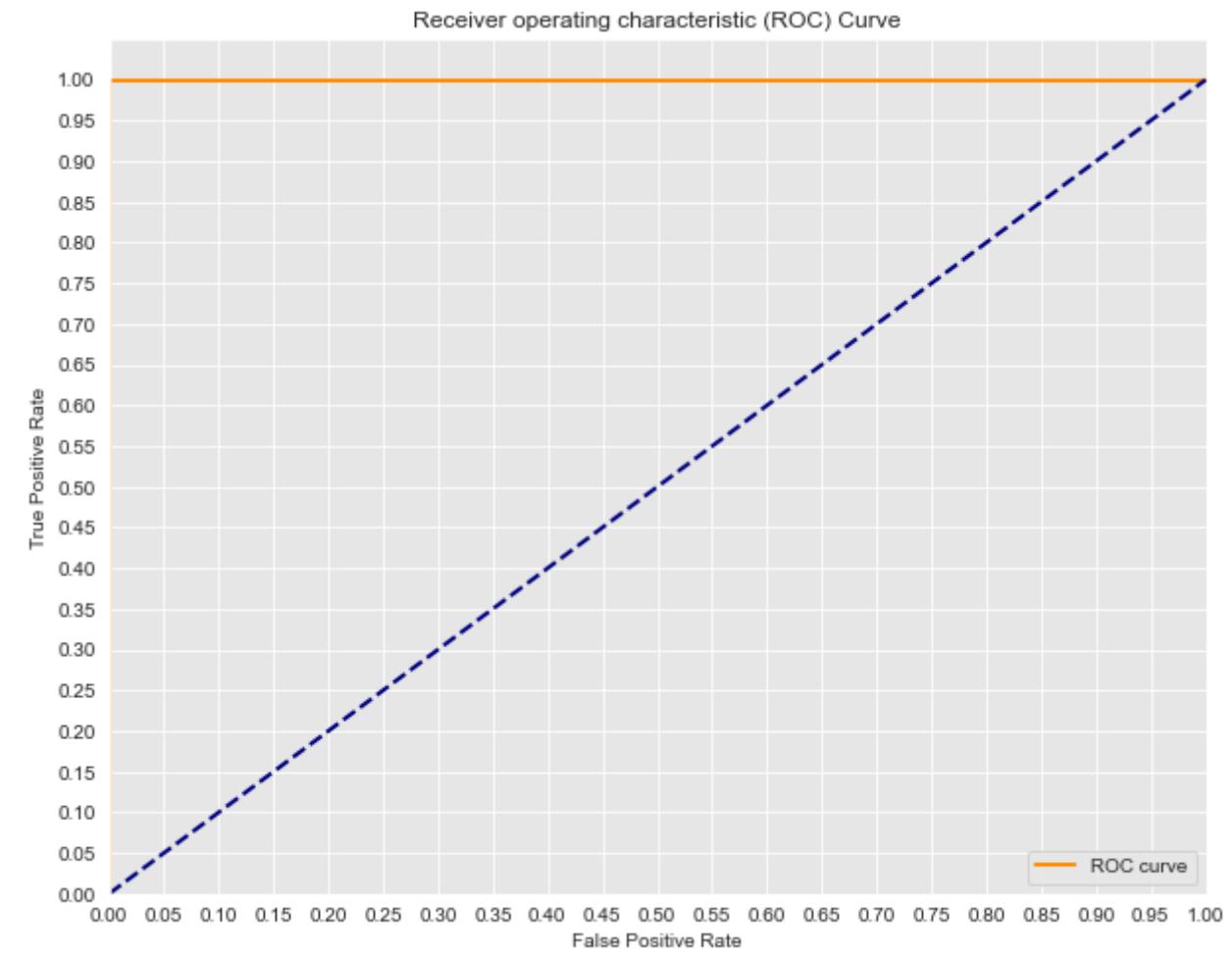
$$ReceiverOperatingCharacteristic = \frac{TPR}{FPR}$$

$$AreaUnderCurve = \int_a^b TPR(FPR^{-1}(x))dx$$

```
In [94]: multiclass(decision_tree,X_train,a,p,translation_df1['mRNA_Codons_copy'])
```

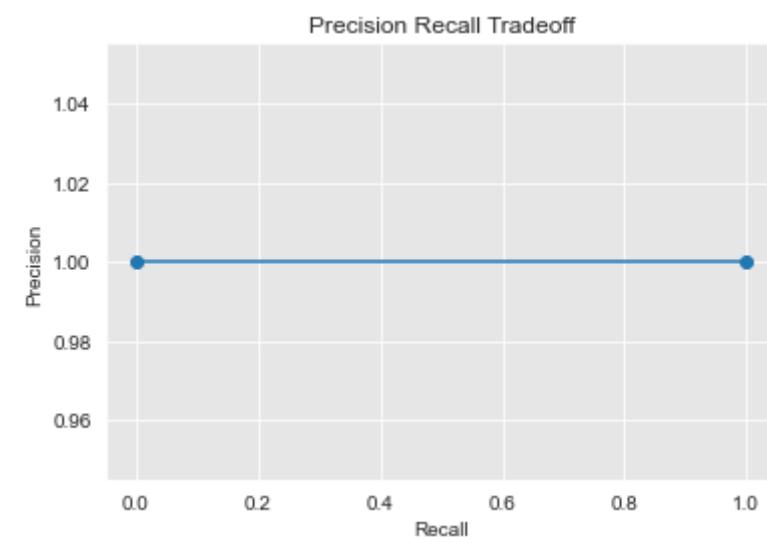
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

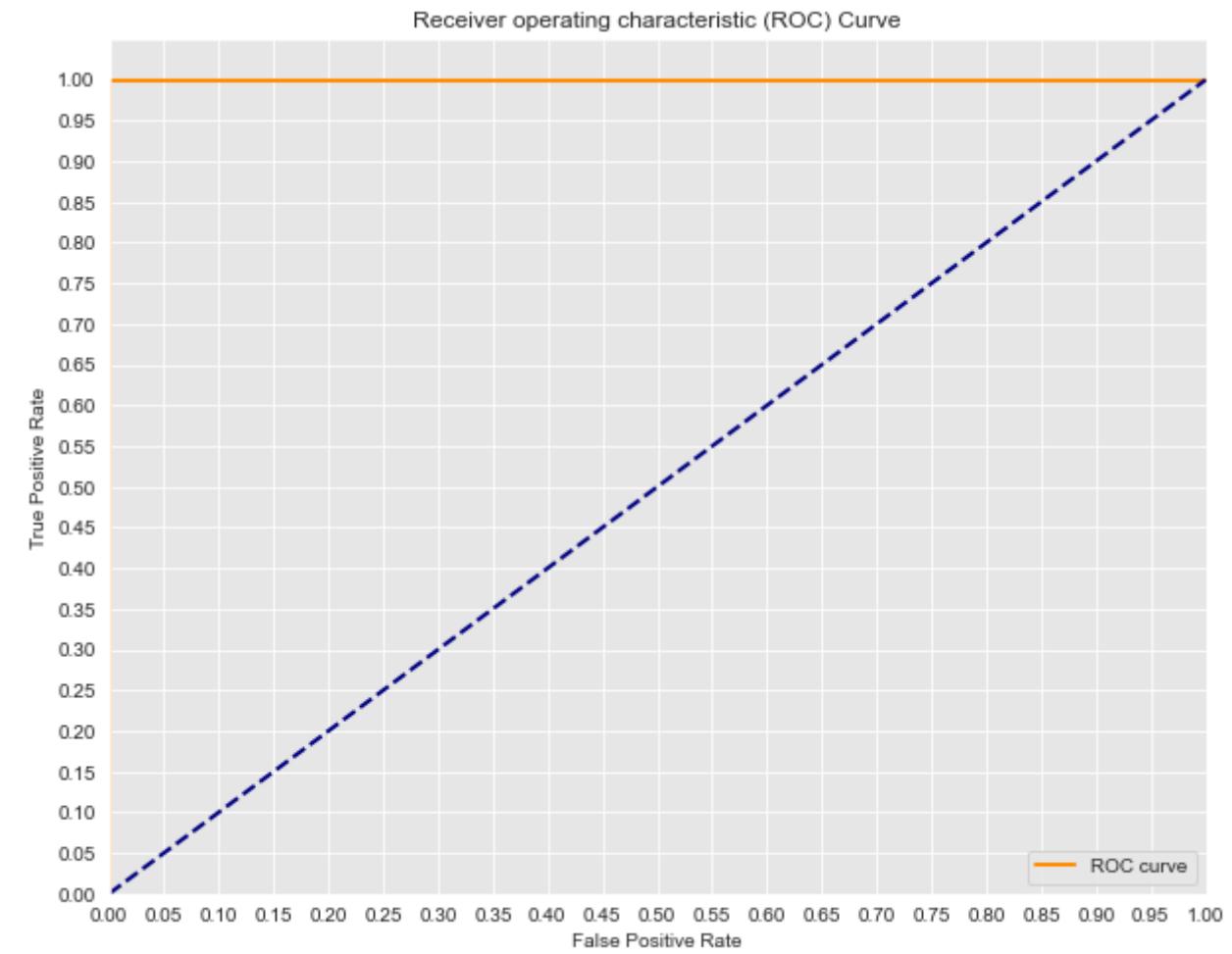




Cross Validated ROC AUC score: 1.0

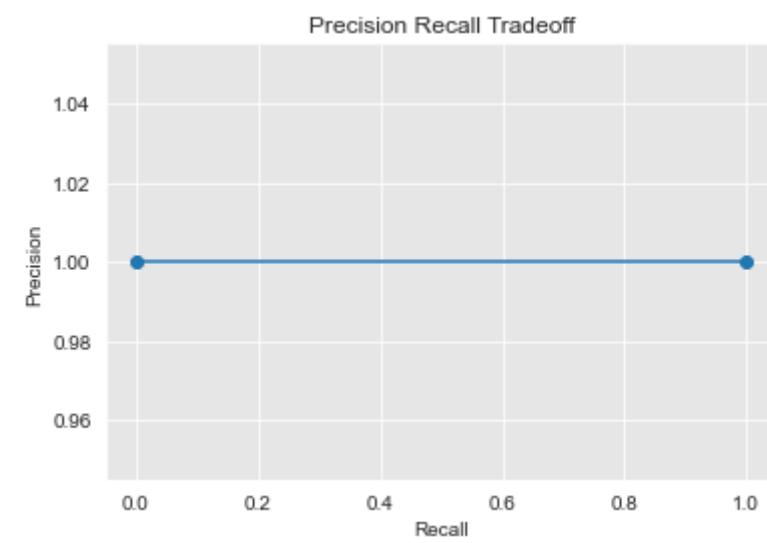
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

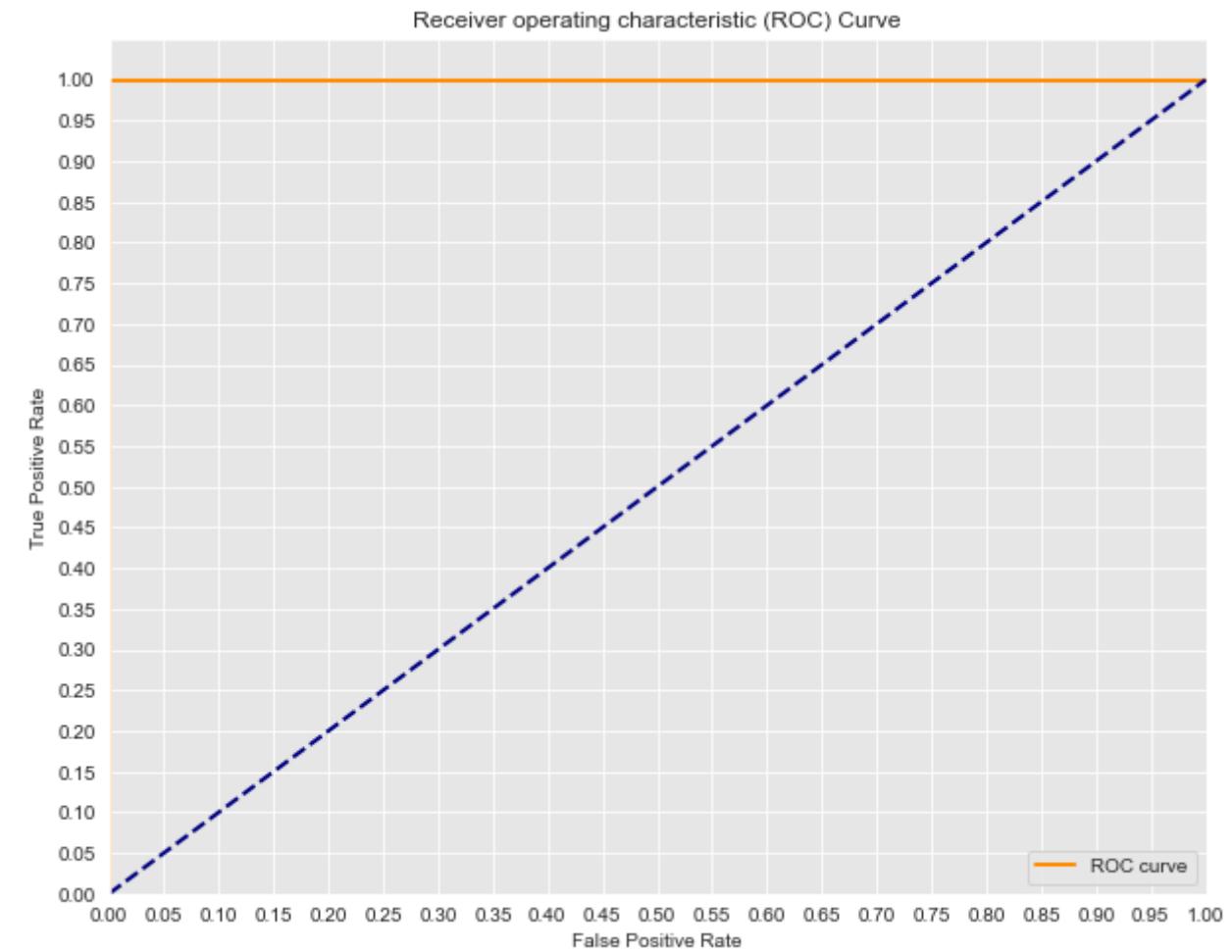




Cross Validated ROC AUC score: 1.0

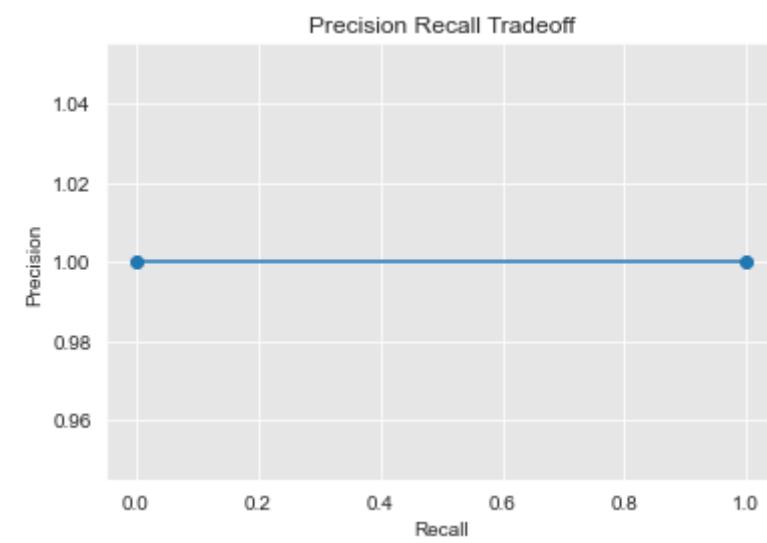
Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

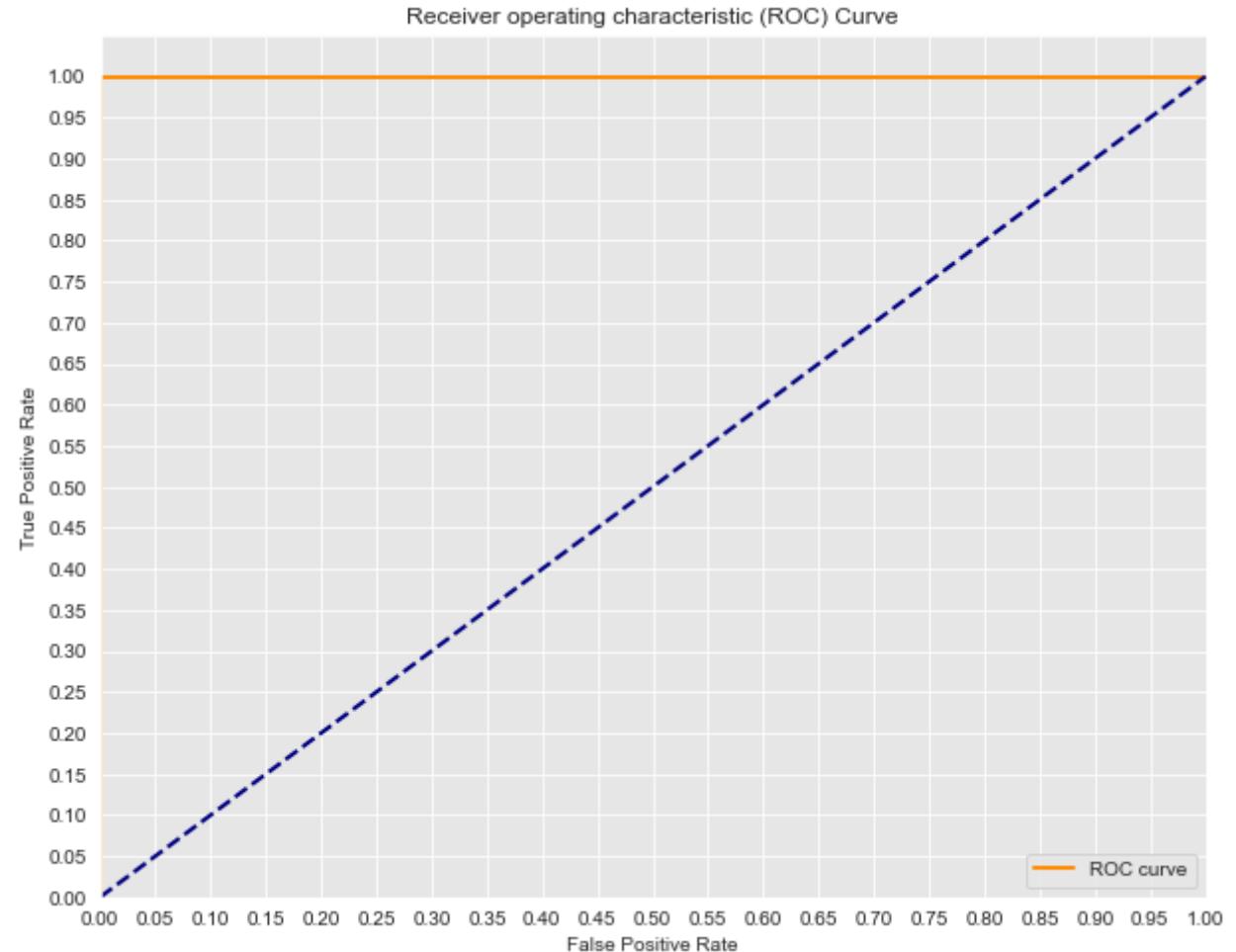




Cross Validated ROC AUC score: 1.0

Class:U
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

Random forests are an ensemble of decision trees that output the average prediction of the trees.

Fit the model to the training data.

```
In [328]: random_forest = RandomForestClassifier()  
random_forest.fit(X1_train, y1_train)
```

```
Out[328]: RandomForestClassifier()
```

Show hyperparameters.

```
In [146]: random_forest.get_params()
```

```
Out[146]: {'bootstrap': True,  
           'ccp_alpha': 0.0,  
           'class_weight': None,  
           'criterion': 'gini',  
           'max_depth': None,  
           'max_features': 'auto',  
           'max_leaf_nodes': None,  
           'max_samples': None,  
           'min_impurity_decrease': 0.0,  
           'min_impurity_split': None,  
           'min_samples_leaf': 1,  
           'min_samples_split': 2,  
           'min_weight_fraction_leaf': 0.0,  
           'n_estimators': 100,
```

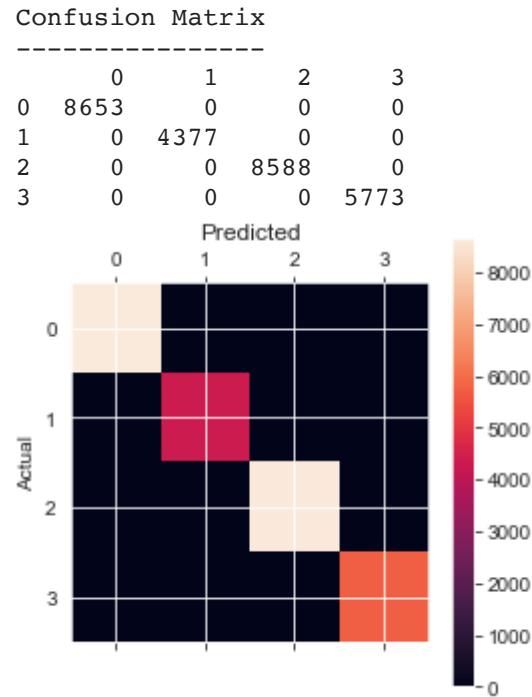
```
'n_jobs': None,  
'oob_score': False,  
'random_state': None,  
'verbose': 0,  
'warm_start': False}
```

Make predictions.

```
In [329]: y1_hat_train_rf = random_forest.predict(X1_train)
```

All predictions were classified correctly.

```
In [330]: con_mat(y1_train,y1_hat_train_rf)
```

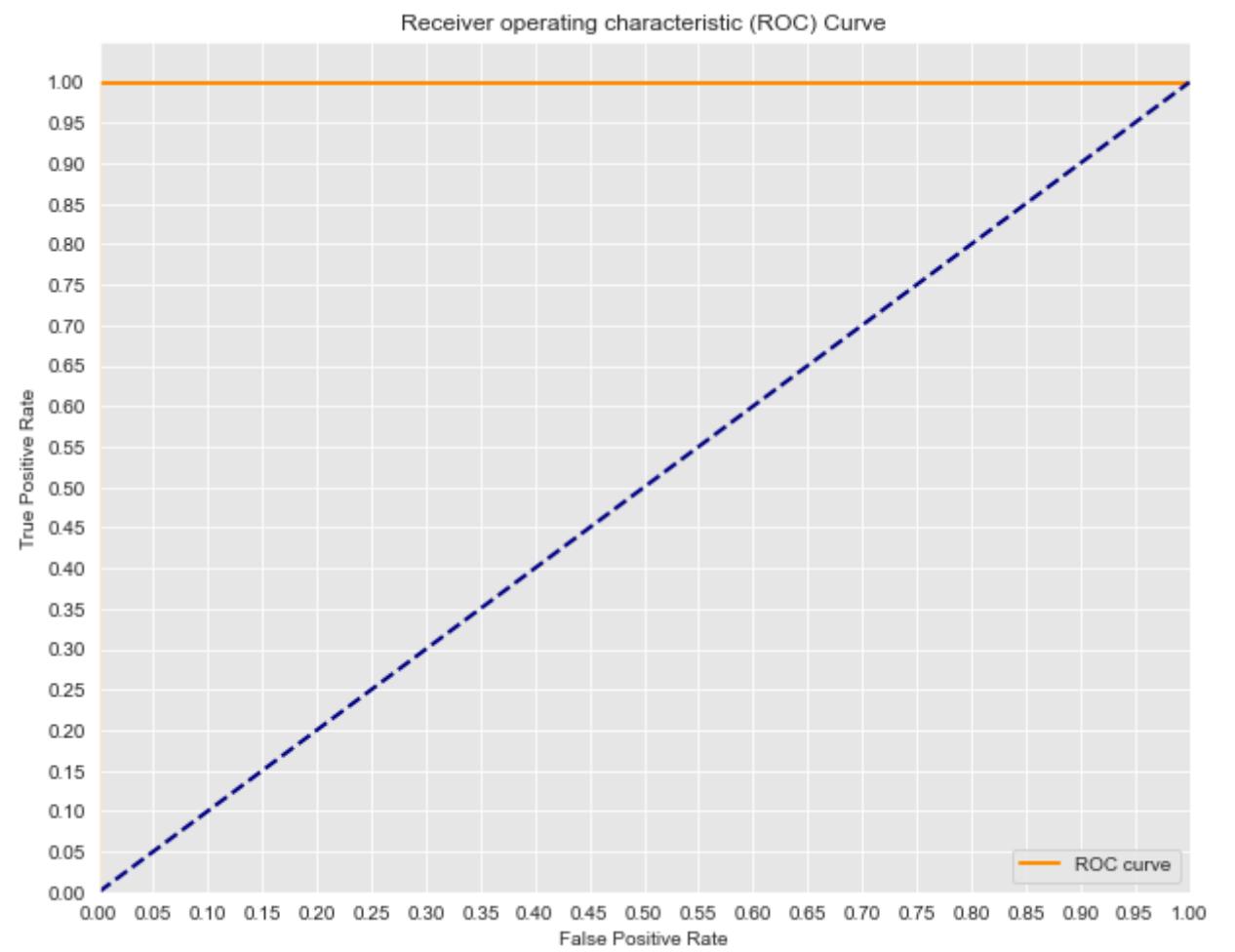
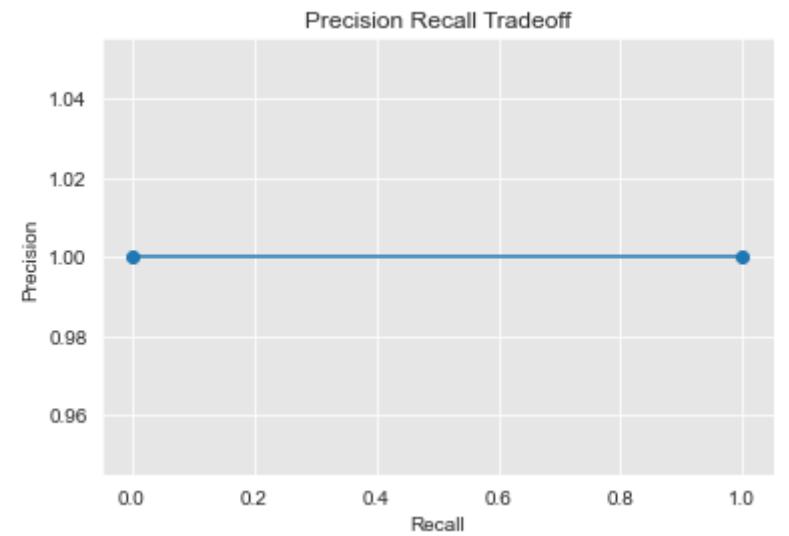


Separate predicted values into each class for metrics.

```
In [98]: predictions_class0=y1_hat_train_rf==0  
predictions_class1=y1_hat_train_rf==1  
predictions_class2=y1_hat_train_rf==2  
predictions_class3=y1_hat_train_rf==3  
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3]
```

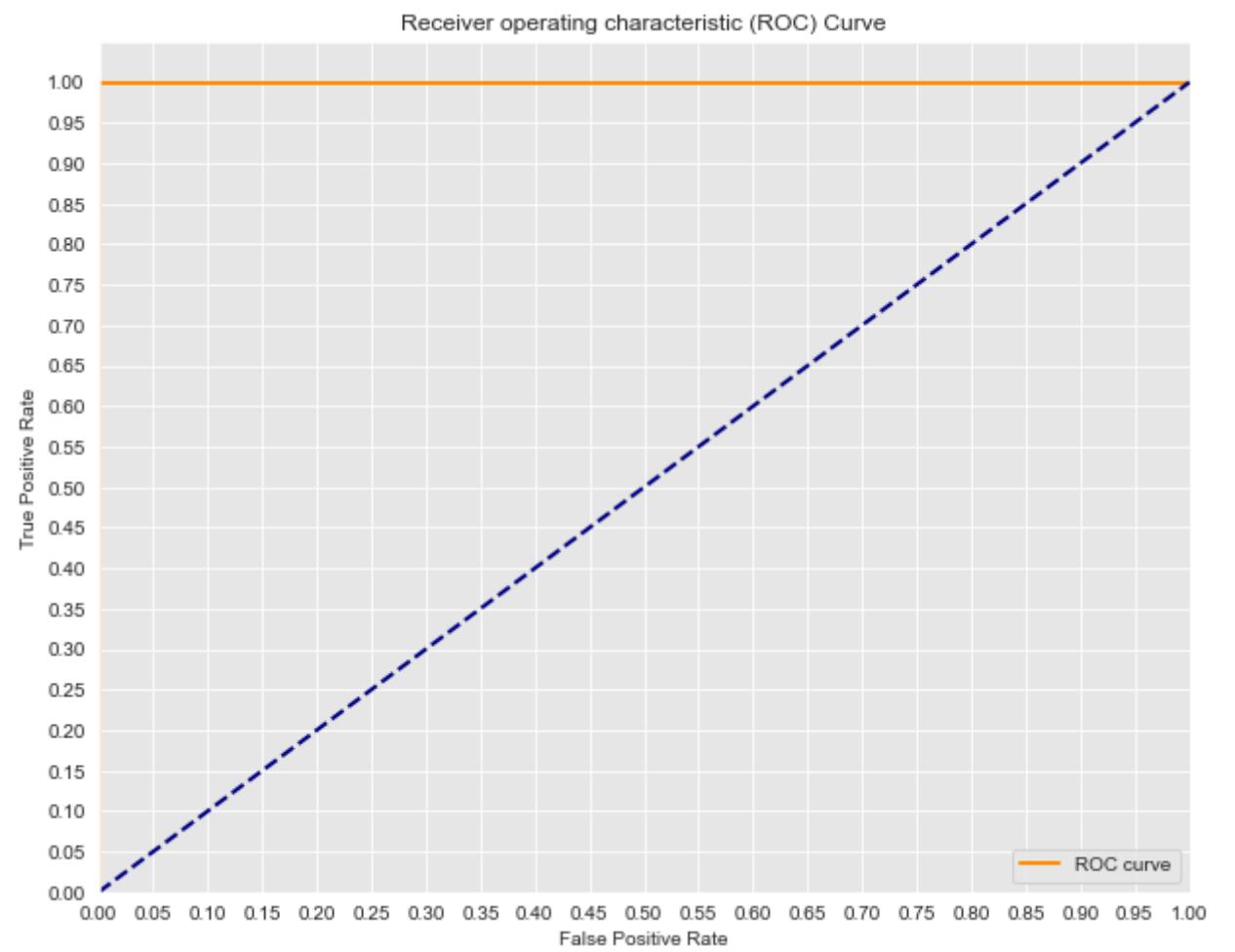
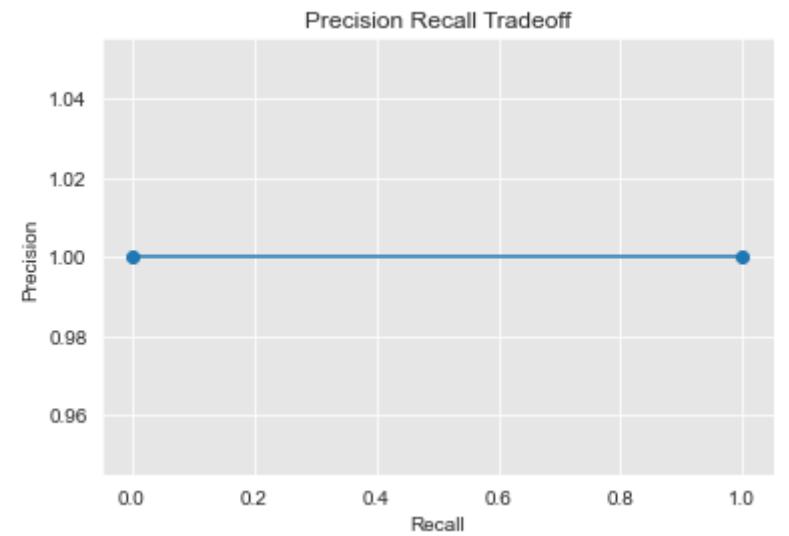
```
In [99]: multiclass(random_forest,X_train,a,p,translation_df1['mRNA_Codons_copy'])
```

```
Class:A  
Precision Score: 1.0  
Recall Score: 1.0  
F1 Score: 1.0  
Accuracy Score: 1.0  
Specificity Score: 1.0  
AUC: 1.0
```



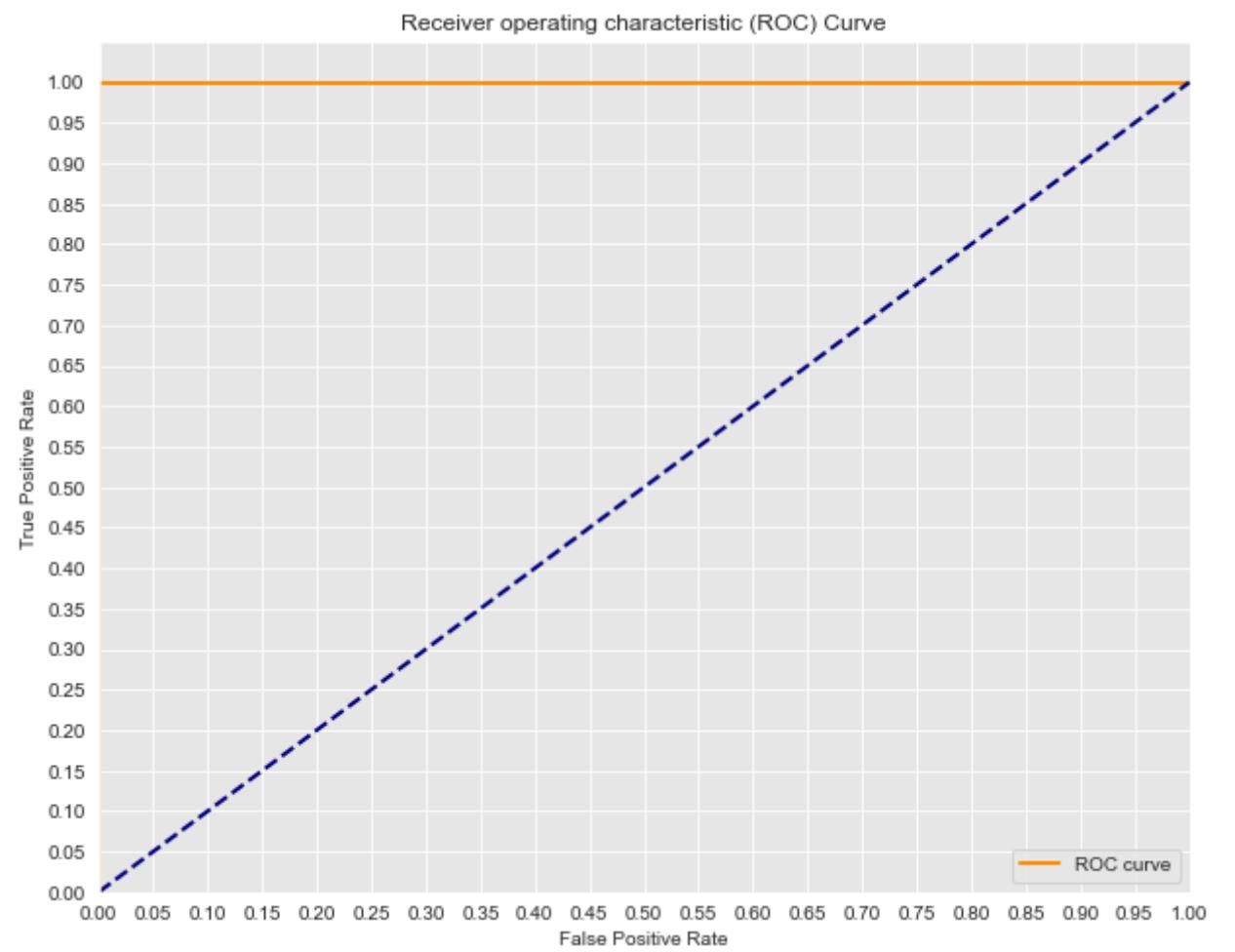
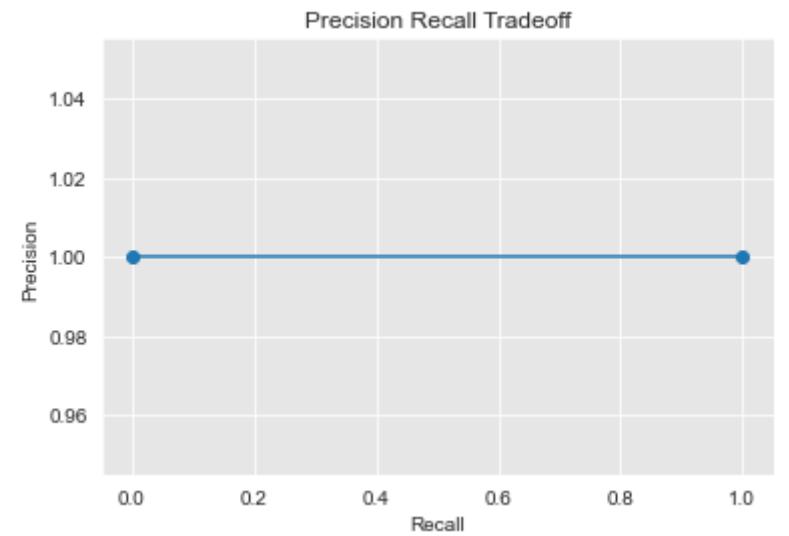
Cross Validated ROC AUC score: 1.0

Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



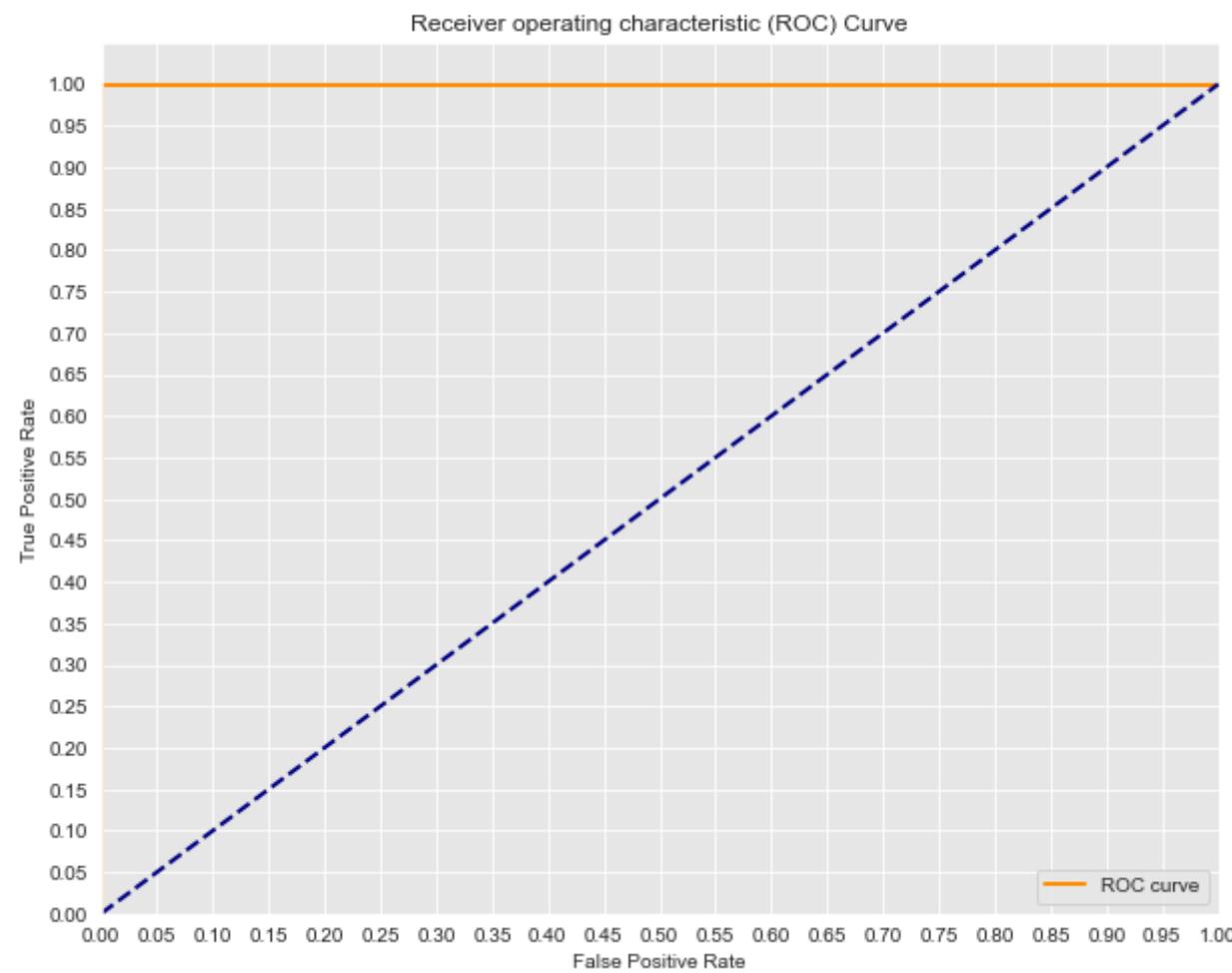
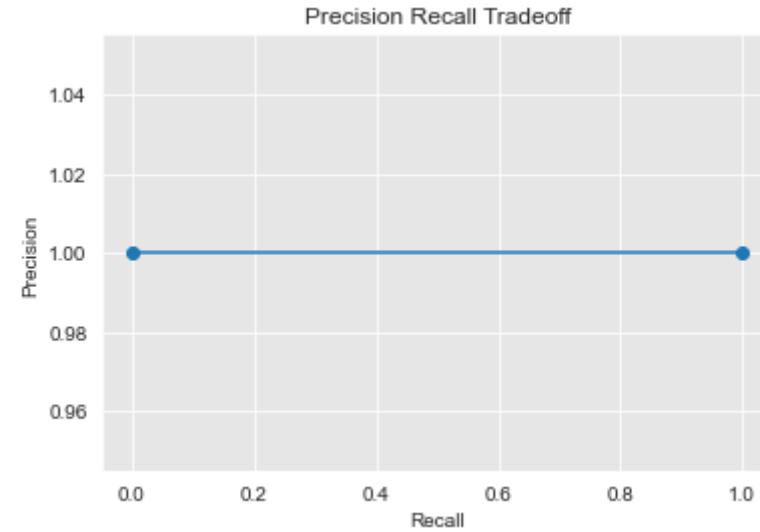
Cross Validated ROC AUC score: 1.0

Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



Cross Validated ROC AUC score: 1.0

Class:U
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



Cross Validated ROC AUC score: 1.0

Gradient boosting is an ensemble of weak learning decision trees that iteratively combine into a single strong learner. Each tree is combined with the previous one in order to minimize the overall prediction error by optimizing a differentiable loss function. $f_0(x)$ is the initial tree, m is the iteration, $\phi_m(x)$ is the current tree, and θ_m is the weight for iteration m .

$$f(x) = f_0(x) + \sum_m^M \theta_m \phi_m(x)$$

Fit the model to the training data.

In [309]: `gradient_boost = GradientBoostingClassifier()`

```
gradient_boost.fit(X1_train, y1_train)
```

```
Out[309]: GradientBoostingClassifier()
```

Show hyperparameters.

```
In [147]: gradient_boost.get_params()
```

```
Out[147]: {'ccp_alpha': 0.0,
            'criterion': 'friedman_mse',
            'init': None,
            'learning_rate': 0.1,
            'loss': 'deviance',
            'max_depth': 3,
            'max_features': None,
            'max_leaf_nodes': None,
            'min_impurity_decrease': 0.0,
            'min_impurity_split': None,
            'min_samples_leaf': 1,
            'min_samples_split': 2,
            'min_weight_fraction_leaf': 0.0,
            'n_estimators': 100,
            'n_iter_no_change': None,
            'presort': 'deprecated',
            'random_state': None,
            'subsample': 1.0,
            'tol': 0.0001,
            'validation_fraction': 0.1,
            'verbose': 0,
            'warm_start': False}
```

The direction of the steepest descent is given by the negative gradient, $-g(x)$, of the loss function, L . At each iteration, a tree is fitted to predict the negative gradient for the next tree, ϕ_m . The negative gradient determines the direction of the step and line search, proposed by Friedman, determines the step length of ρ with an η learning rate.

$$L_{MSE} = \frac{1}{N} \sum_i^N (y_i - \hat{y}_i)^2$$

$$-g_m(x) = -\nabla \frac{\partial L_{MSE}}{\partial f(x)}$$

$$\phi_m = \operatorname{argmin} \sum_i^N [(-g_m(x_i)) - \phi_{m-1}(x_i)]^2$$

$$\rho_m = \operatorname{argmin} \sum_i^N [((-g_m(x_i)) + \phi_m(x_i)) - y_i]^2$$

$$f(x) = f_0(x) + \sum_m^M \eta \rho_m \phi_m(x)$$

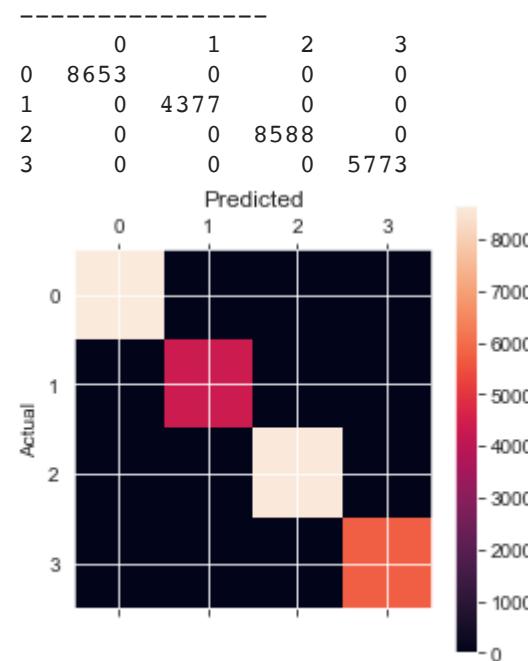
Make predictions.

```
In [316]: y1_hat_train_gb = gradient_boost.predict(X1_train)
```

All predictions were classified correctly.

```
In [326]: con_mat(y1_train, y1_hat_train_gb)
```

Confusion Matrix

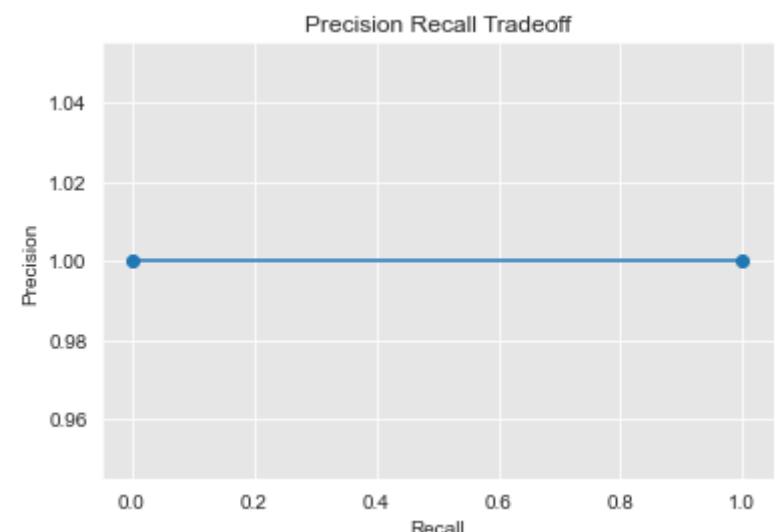


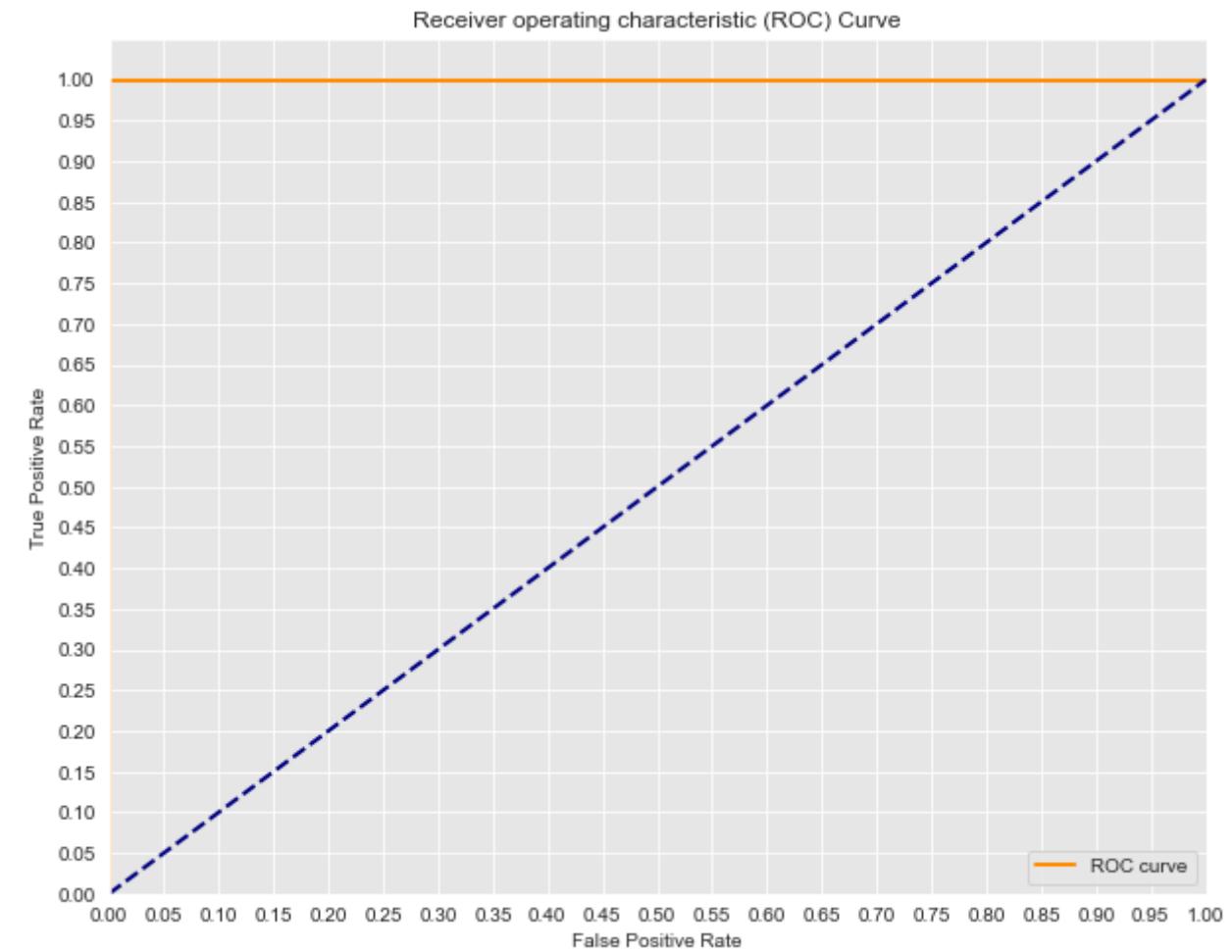
Separate predicted values into each class for metrics.

```
In [118]: predictions_class0=y1_hat_train_gb==0
predictions_class1=y1_hat_train_gb==1
predictions_class2=y1_hat_train_gb==2
predictions_class3=y1_hat_train_gb==3
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3]
```

```
In [120]: multiclass(gradient_boost,X1_train,a,p,translation_df1['mRNA_Codons_copy'])
```

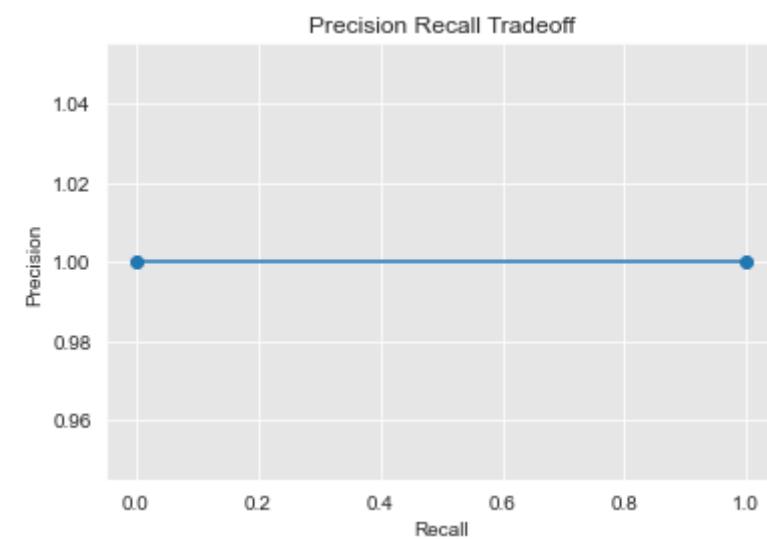
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

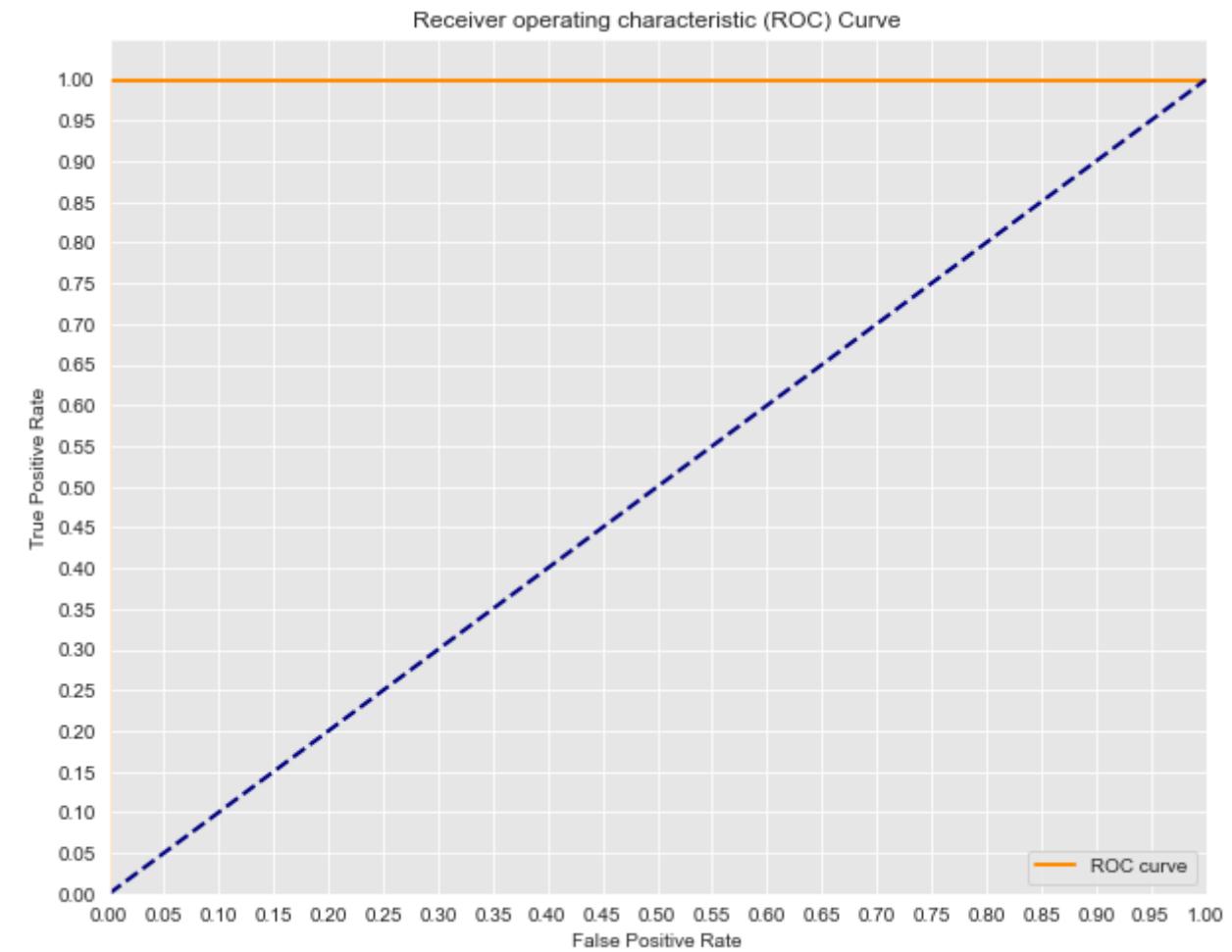




Cross Validated ROC AUC score: 1.0

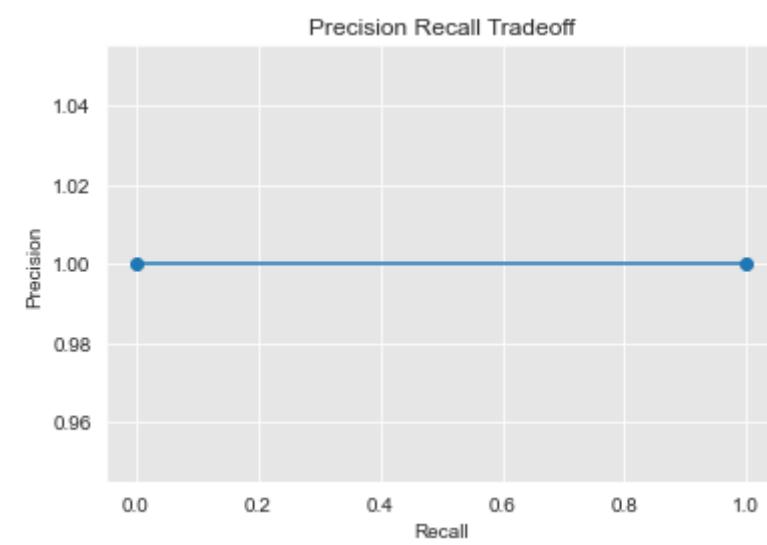
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

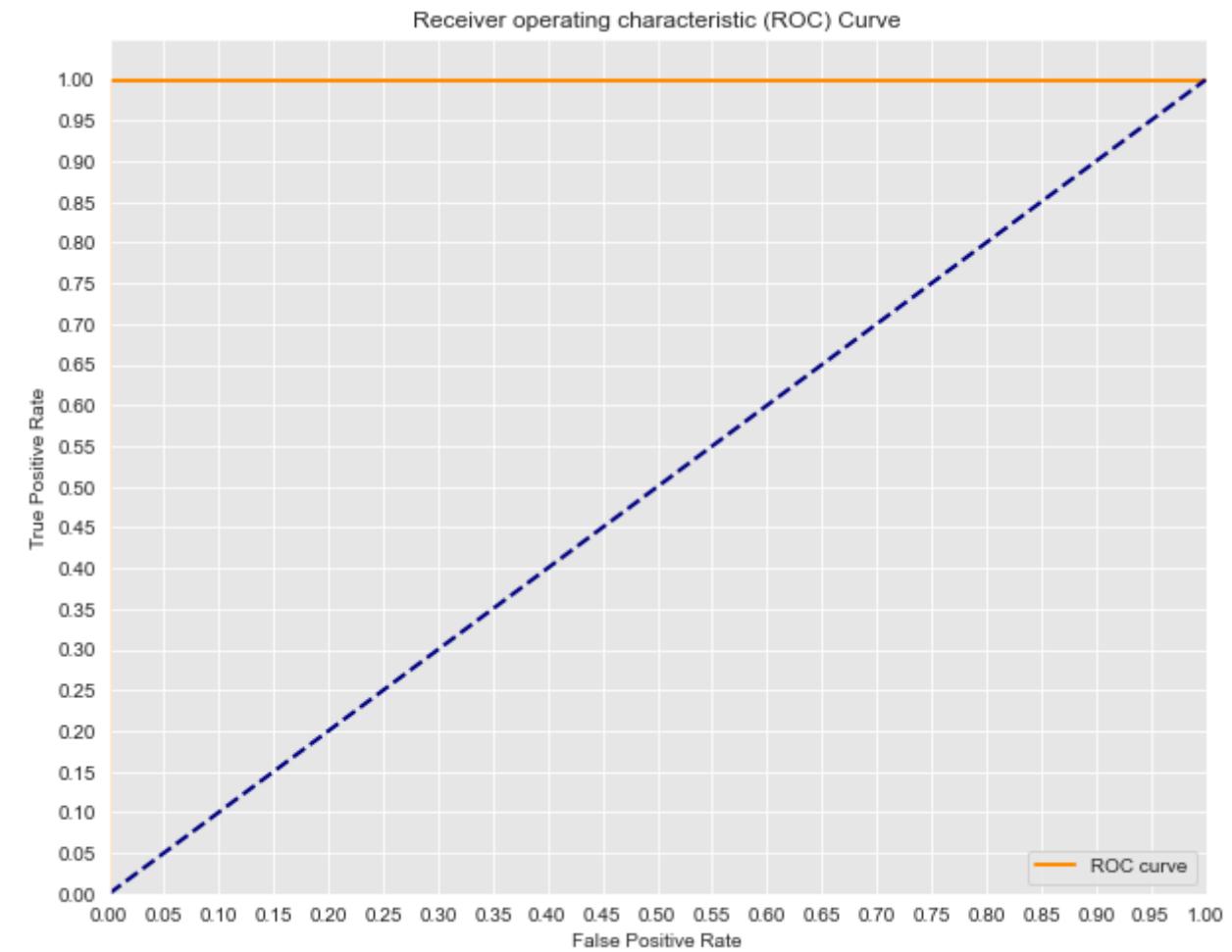




Cross Validated ROC AUC score: 1.0

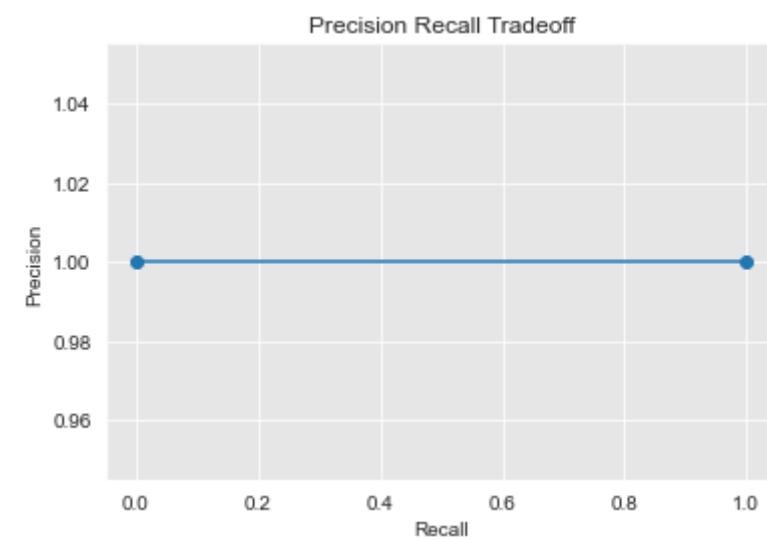
Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

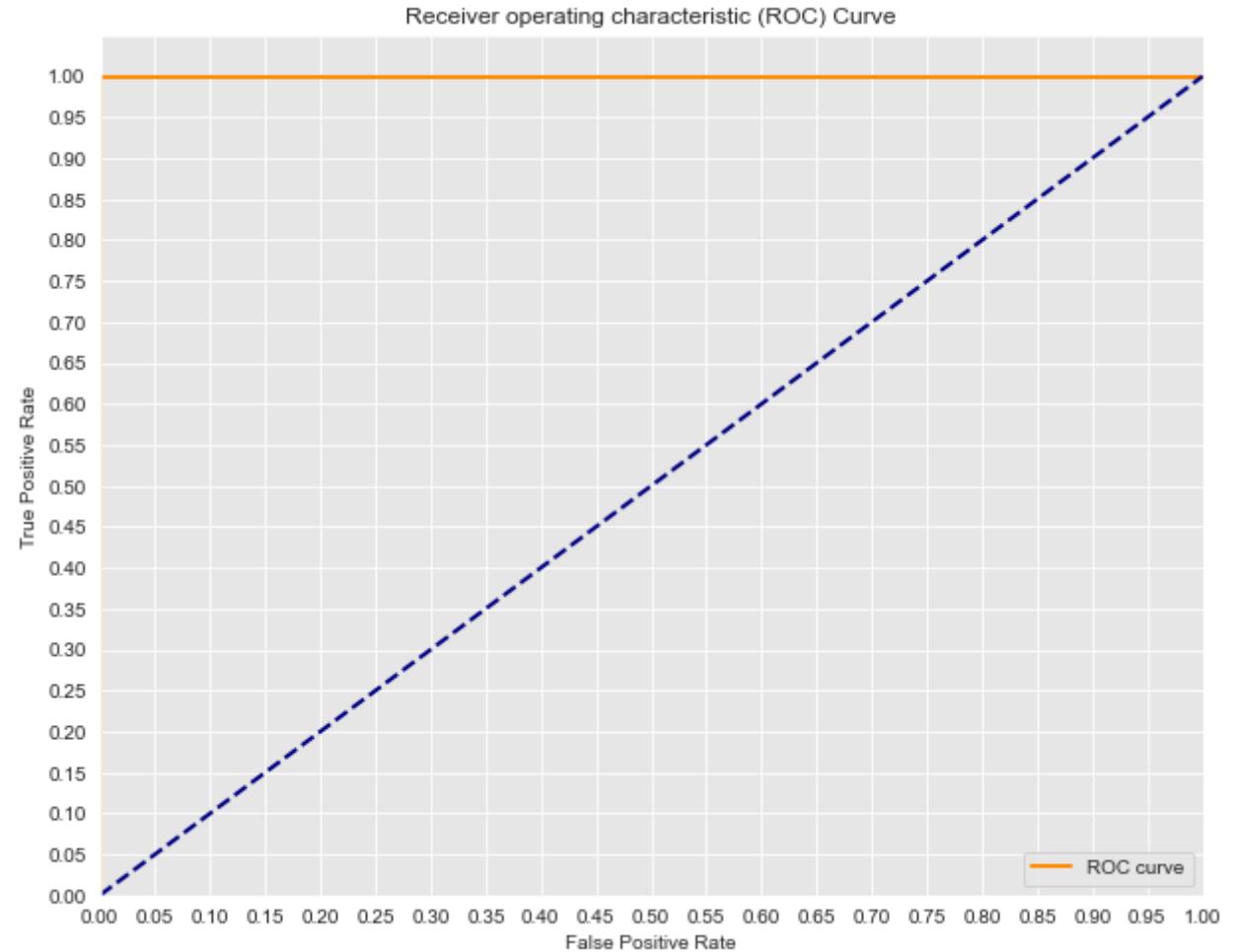




Cross Validated ROC AUC score: 1.0

Class:U
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

All models performed with a cross validated roc auc score of 1. Will test on gradient boost model.

Fit the model to the test data.

```
In [311]: gradient_boost.fit(X1_test, y1_test)
```

```
Out[311]: GradientBoostingClassifier()
```

Make predictions.

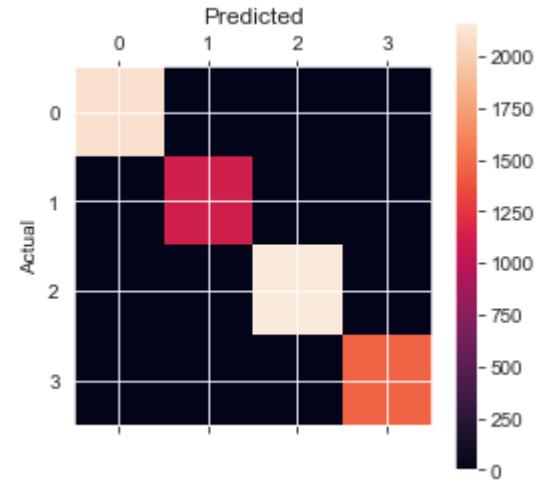
```
In [313]: y1_hat_test_gb = gradient_boost.predict(X1_test)
```

All predictions were classified correctly.

```
In [331]: con_mat(y1_test,y1_hat_test_gb)
```

Confusion Matrix

	0	1	2	3
0	2100	0	0	0
1	0	1121	0	0
2	0	0	2165	0
3	0	0	0	1462

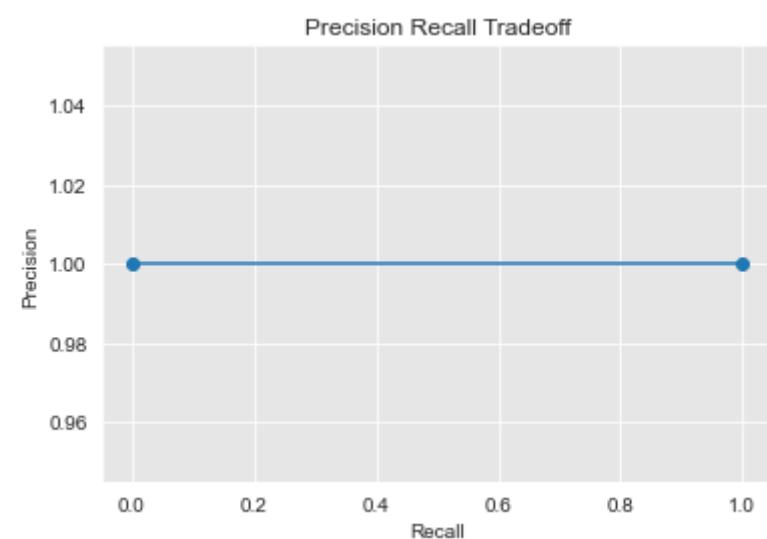


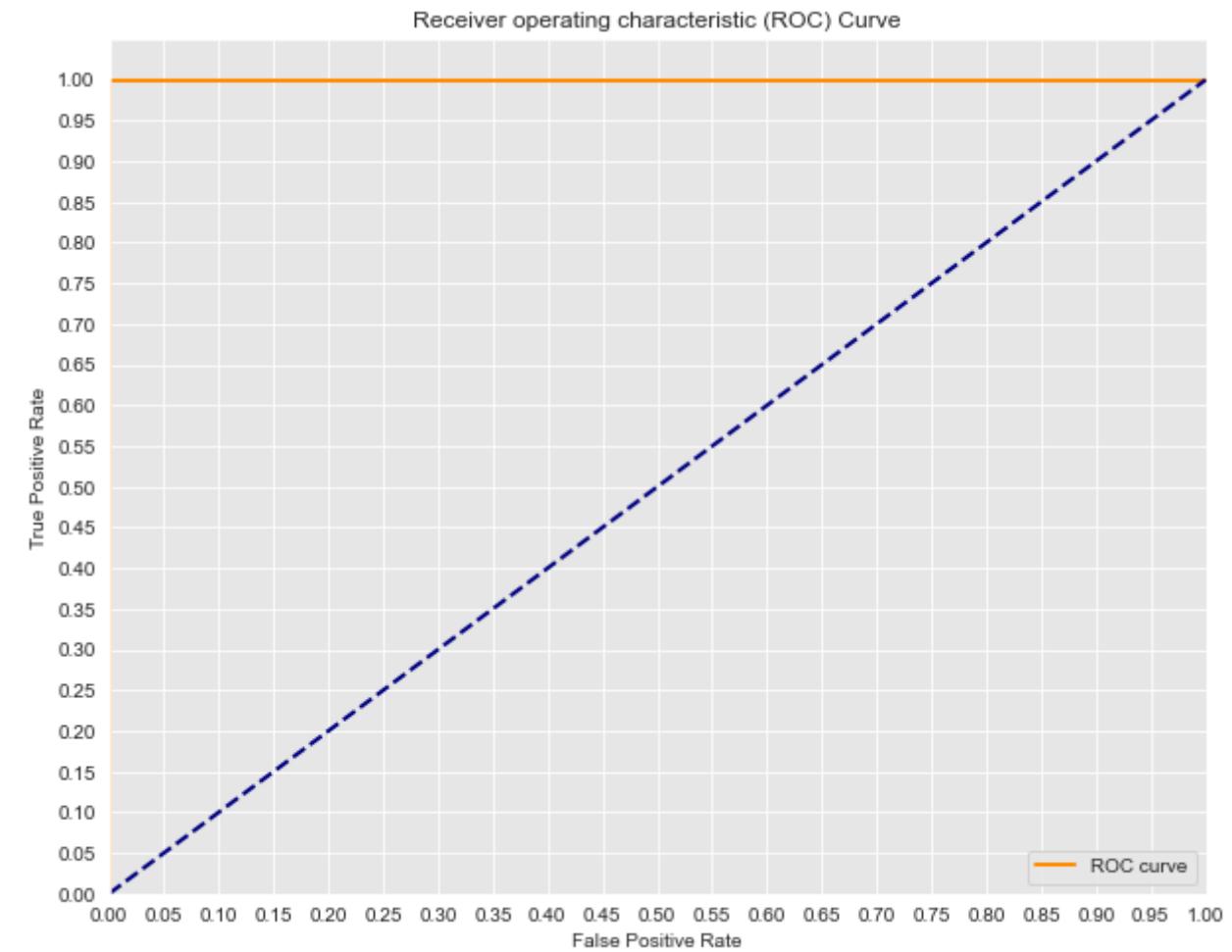
Separate actual and predicted values into each class for metrics.

```
In [322...]
actual_class0=y1_test==0
actual_class1=y1_test==1
actual_class2=y1_test==2
actual_class3=y1_test==3
predictions_class0=y1_hat_test_gb==0
predictions_class1=y1_hat_test_gb==1
predictions_class2=y1_hat_test_gb==2
predictions_class3=y1_hat_test_gb==3
a=[actual_class0,actual_class1,actual_class2,actual_class3]
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3]
```

```
In [324...]
multiclass(gradient_boost,X1_test,a,p,translation_df1['mRNA_Codons_copy'])
```

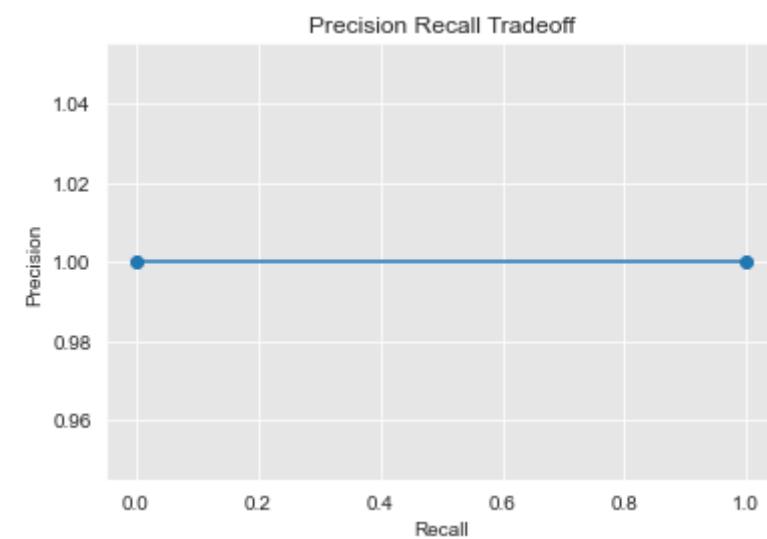
```
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0
```

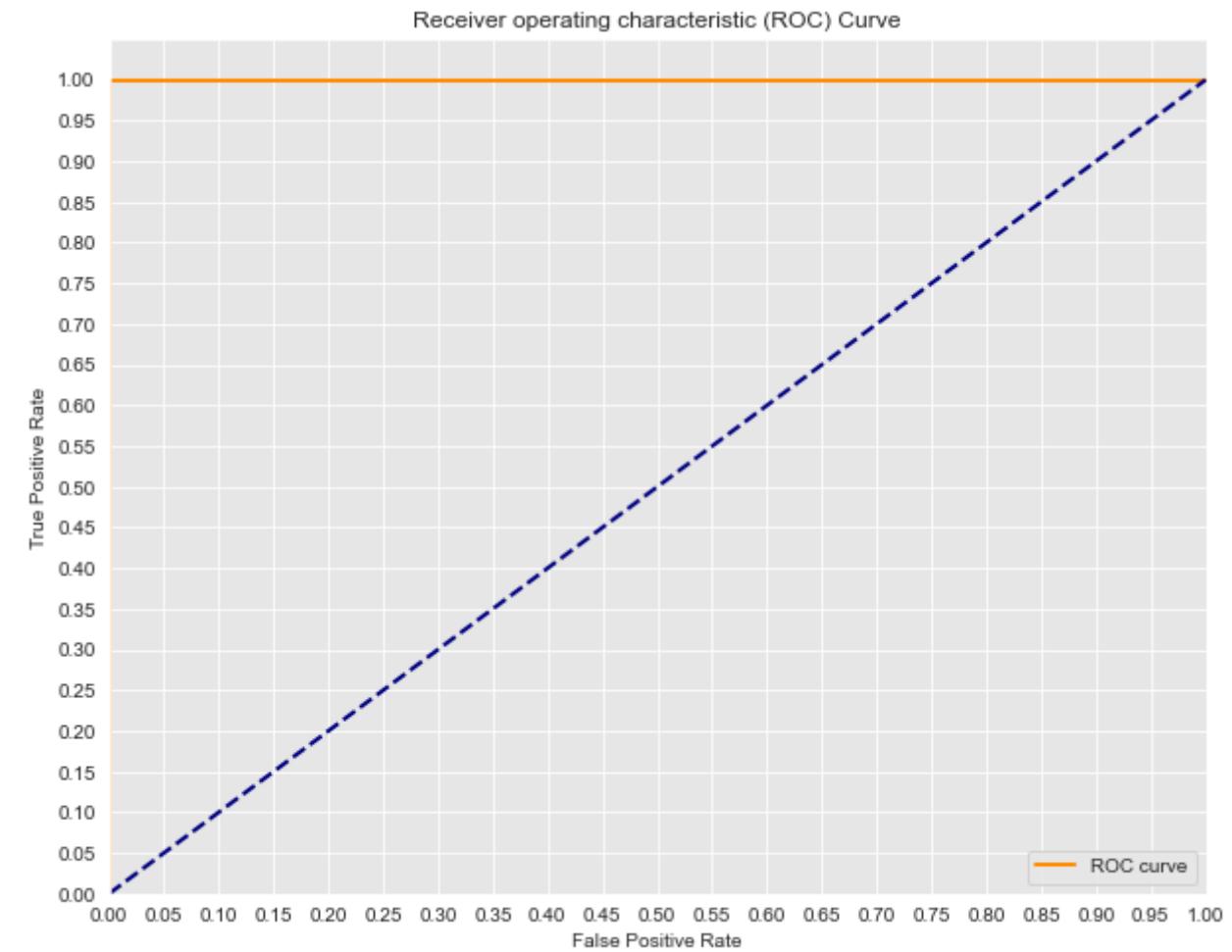




Cross Validated ROC AUC score: 1.0

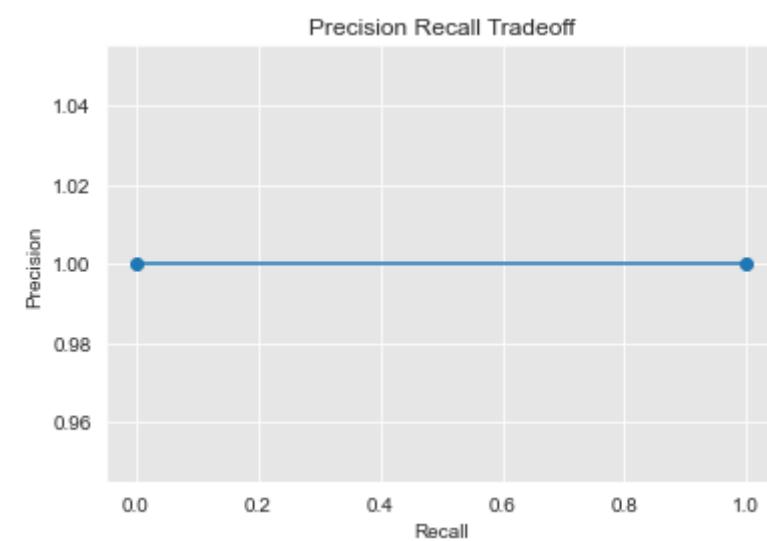
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

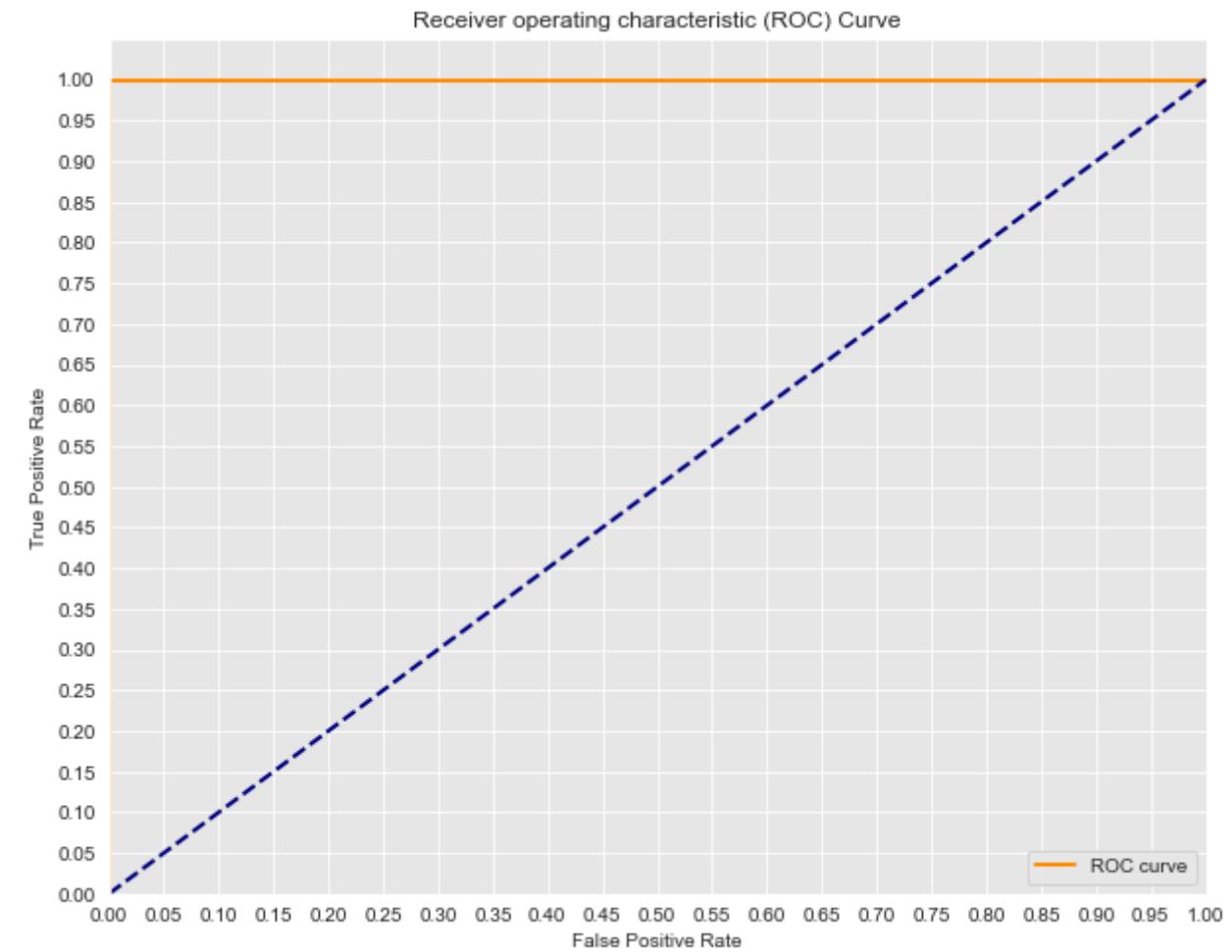




Cross Validated ROC AUC score: 1.0

Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

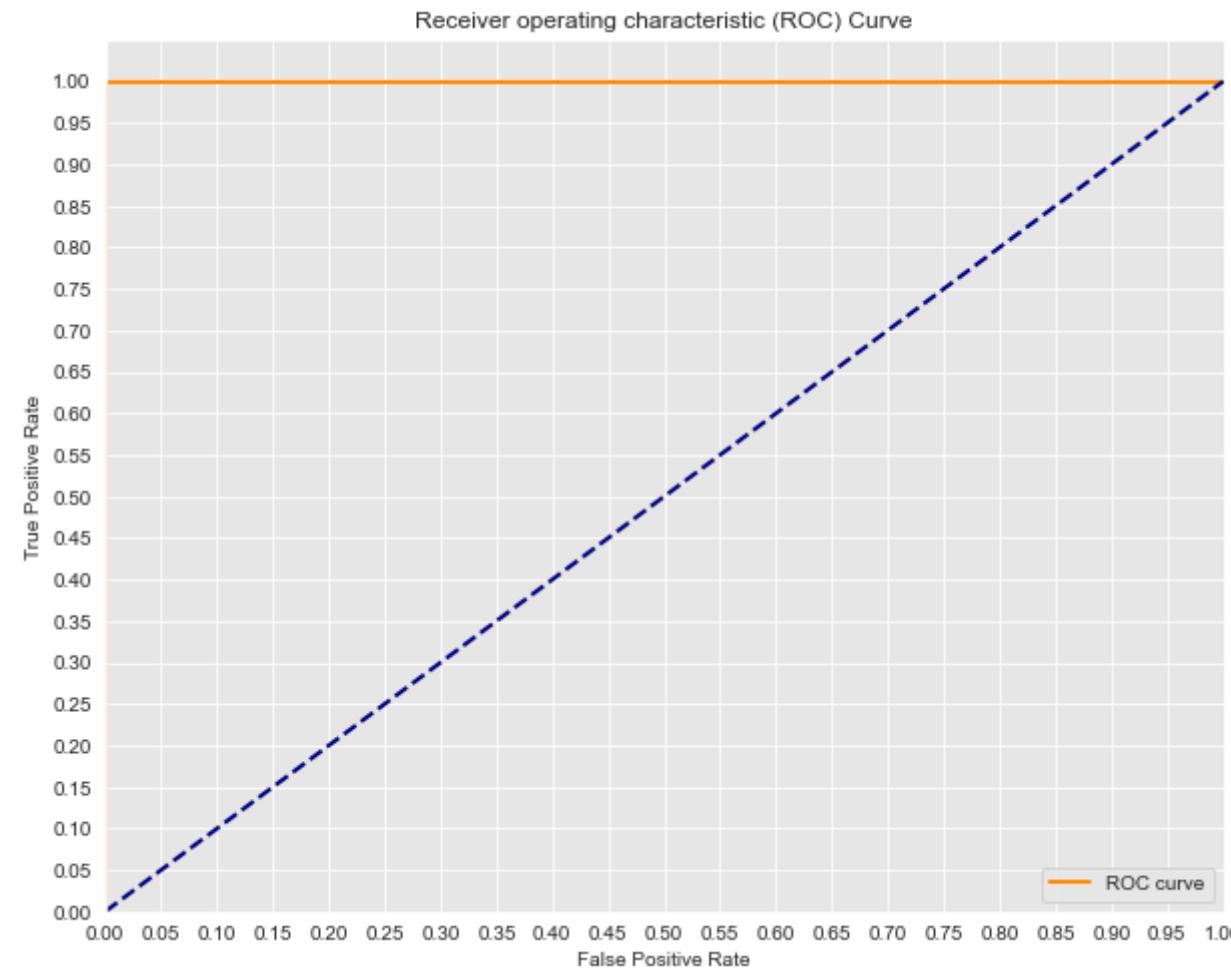




Cross Validated ROC AUC score: 1.0

Class:U
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

Next transfer anticodons to amino acids.

Data Cleaning

Take predictions from previous model and change them back to genome letters.

```
In [336]:  
c=''  
for i in y1_hat_train_gb:  
    if i == 0:  
        c+= 'A'  
    elif i == 1:  
        c+= 'C'  
    elif i == 2:  
        c+= 'G'  
    elif i == 3:  
        c+= 'U'  
  
c
```

Out[336...]: 'UACGACAGGAAUACAUAAAUGGGGGAAUCGGUUCGUAGGUAAUACCGUUACGCAGGUUCGGCAUAGGUAGCUGGAUCAGCGAUGGUCCUAAGAUAGAAGGAAUAACAGAGAAGGUCAUGGGACGCAAUUCAUCCCCCGACAAUCAUACGGCGAAGGC AUGCAAGGGUAUAGGGAAAUAGUGCCGCAACGAAGCGAAGUGUCGGCUCUAGCGUUGAGAACGGGUAGCAAGCACGAGUAAGACCACAAGUGAAAUGCAGGUUAAAGCAGUUGGAGCAGUCGGCAAUGCACAAAGGGACCUAACAGCUACAAAAGGAACAAUGAGCAAG GCAUAACAAGACAUAAUGUGUCUCGAUUGAACGUAGUUCUAGCUACUCAGGAGGUACGGCAGGUUAGCUAGGUAGCAGUGGACCGAAGGCACCGCUAGUACGUGGGUGGGAGUAAAGGAAGCUCGGAUUCCCGGUAAAACUCUGGU GAGAAAGGCGGACCUAGACUAGAGGGCAAGGACUGGUACCGCGGAGGGACGUAGUACGUUAGAGCGACGUAAAUCGGGAGGCAUUUUGGUAAAAUAGUAUAACCAUUUGAAUCUGGAAUAGUGCUGGGCCAUGGGAGGAGCGAAGGCCGAUC CGGGAGUAGCUAUAGGAAGGAGGAGCGAAGGCCGAUCGGGAGACCAAGUAAAAGAAGAACGC AUAGAAAAGUAAGUACAGAUGCACGAAAGAUAAUAGGUAGAAGCUGGUAGACAGGGGUGCCAAUGACUGAGGAUCGAAGUCGUAGCAAGUACUGAAGGUGUAAAUGCAACCACUUGUGGAAGAAUUAUGAGAAUUGCUCGAGAAUACUGUC UAGAACUAACGAAUACGGACGCAUGGCGAGAGGGCUGGUAGCGACGAUGGAAUAAAAGGGAACUUAAGGUACGGAGGAGGAAGGAGGGACAGCAGAACUCUAAGGGGUACGGUAAAACCUUAAGGUCAGCCUAAGGUGGGAGGCUCAGUGCCAAACGAGA GAAGUAGCAUAGCUAAACGAGAGAAACGUUAGGGUGACUAUAGGAGCGAAGGGAAUGUACUAGGACGACGGAGGUAAAAGGGAGAGGUUAGGAGGACCCAGGGACGCCCCGGGAAAGGGGAGAGGUUAGGACAAAGGUUAAGGAGAGGUUAUCGCAUAGGAAUUCAGUGAU

AGGAAAGAGCGUGGGGGAGAUGCACGGGCCACGCCAUAGAUGGGCGAUAAAUGCUGGUUCGGAGUGAGCGACGCUGAGGGGGGGAGAAAAAUUCGACAAAGCAACCUCAGAGAAGUCUCGAGCCUCGAUCGGUCAAACAGAGGGGUUGCCUGGGUGCUAUCUCUCAUGUGGU
GGACAGAACAUUUCCUAGGCCAACGACACAUUAGGACCAACAGAACAUAGGACCUACAAGAUGAAGGUGGUGGCAUGAGGAGGGCCAGUCGGGAAAGUAUCACACAAACGUAAAAGGUAGGGGAGAAAGUAGGGUAGGAGAAUUGGUAAAACGGCCUUUACAGAGCACUGCAU
AGAAGCCUAAGAGCUGAGGUUGCAGGGGUAAACGUAAGUAGAACCAGGGGAAACAGAAAGAGAUAAAUCUAAGGCAGGGCAGUAAUAAAGCAGUGGCCAGAGGACGGAAAUAGAGGAAGGAGAAGGUAGGAGGUACAGGGCAGGGGAAAGUAGGGGAGAAUUGGUAGGAGGUACGGGACCCACACUAGGACAGGC
GGGUUUUAAGAGUAUCAAAGAUGAGAGCAGGUUUUAAACAGAACAUAAUAAAAGAUGAGGGAUUCCAGCAAAGUUCGUAGGGUGGGCGCACGGUCAGGGGAAAGUAGGGGAGAAUUGGUAGGAGGUACUGGAGGUUUAUGUGACAUUGGGUGCGCUGGAAAAAUUCUU
AGGUUAAGGACGAGUGCACUAAAAGACGAGGUUGGCGCGGCUUGACAAGUGGGGACUUGGUACACAGCAUUGCGUGUAACGUUUACGUUCAGUACAAUUGCUACUAUCCAAUUGGAGGACGGAGUCCAGAUACACUAGCAGUGAAGCUGAUGUAGGGAGAUGCUAUCGUG
CUCCGGAGACAUGCGAAGGGACCAAGUUUACGGGCCGAAUUCGGGACCUAUGGAGAUAAAAGUGAAGGUGCAAGGGGAGGUACUGGGGUUGCGAGGAGGGGAAAGUAGGGGAGGUACGGGAGGUACGGGUUGCGAGGAGGGGAAAGUAGGGGAGGUACGGGUUGCGAGGAGGGG
GCCCUAUUGGAAGUAUGCAGGUAGAAGCUGUGCAGUGGAAGUUGCGAGGAUGCAGUGGUGACGAGUAGCGUACACAGGCUGGUCCAGAGAUGGUCCACAGCCCCUAGGAAUUGGCACCGGUGACGUGGUAGGGUUGCGAGGCUUAGAAAACAAAAGGGCAAGGAAAUAACGGCUUCG
AAGGCCAAAGUGCGAAAGAACGGAGGUAAACAUUACUUUAGAUAGCUAAACACAAUGUACGAUGCAAGGAAUUGGCGAUGGUGUAUGGGGGGUAGGUAGGAAAGGUAGGUACGGGUUGGGGUAGGUAGGAAAGGUAGGUACGGGUUGGUAGGGGUACGGGUUGGUAGGGGUACGGGU
CUAGAACUCAUAGUACGAAAGAGACGGACGGUACUCGUGCGACAAGGAAUAGGUAGACAAAGUCCAGCAAGAGAACUACGUACAUUGUGUGGGGUAGGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGU
UCAAGCACCGUAGACCACAGGGUCGGAACCAAUUUGGUGAAAAGAGGUACGUACGAGGUACGCCAAUUGUUACGAACUAGGGUGUAGGUACUUGGGAAUUGGUUGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGUACGGGU
AGGCCACAAGGC'

Break anticodons into threes.

In [337...]

```
anticodons = []
for i in range(0, len(c), 3):
    anticodons.append(c[i : i + 3])

anticodons
```

Out[337...]

```
['UAC',
 'GAC',
 'AGG',
 'AUA',
 'UAC',
 'AUU',
 'AAU',
 'UUG',
 'UGG',
 'GGA',
 'AAA',
 'UCG',
 'GUG',
 'GGG',
 'GAU',
 'UCU',
 'AGG',
 'UAA',
 'UAC',
 'ACG',
 'UUU',
 'ACG',
 'CGA',
 'AUU',
 'GAG',
 'GUU',
 'CGG',
 'CAU',
 'AGG',
 'UAG',
 'UCG',
 'GAU',
 'CAG',
 'CGA',
 'UGU',
 'CCU',
 'AAG',
 'AUA',
 'GAA',
 'GUG',
 'AAU',
 'AAC',
 'AGA',
 'GAA',
 'GGU',
```

'CAA',
'UGU',
'GGG',
'ACG',
'CAA',
'AUU',
'UCA',
'UCC',
'CCC',
'GAC',
'AAU',
'CAU',
'AUC',
'GGC',
'GAA',
'GGC',
'GAU',
'GCA',
'AGG',
'GUA',
'UAG',
'GGG',
'AAA',
'UAG',
'UGC',
'CGC',
'AAC',
'GAA',
'GAA',
'GCG',
'AAG',
'UGU',
'CGG',
'CUC',
'UUA',
'GCG',
'UUG',
'AGA',
'AAC',
'GGU',
'UAG',
'CAA',
'AGC',
'ACG',
'AGU',
'AAG',
'ACC',
'ACA',
'AGU',
'GAA',
'AAU',
'GGC',
'AUG',
'GCA',
'GAG',
'UUU',
'AAC',
'GUU',
'GGA',
'GCA',
'GUC',
'GGC',
'AAA',
'UGC',
'ACA',

'AAG',
'GGC',
'ACC',
'UAA',
'CAG',
'CUA',
'CUA',
'AAA',
'GGA',
'ACA',
'AUG',
'AGC',
'AAA',
'GCA',
'UAA',
'CAA',
'GAC',
'AUA',
'UGU',
'GUC',
'UCG',
'AUU',
'GAA',
'CGA',
'UAA',
'GUA',
'GUU',
'CUA',
'GCU',
'ACU',
'CAC',
'GAA',
'GUG',
'UCA',
'GGC',
'AGC',
'CUA',
'GAA',
'UCA',
'UGA',
'ACC',
'UUG',
'GCU',
'GUU',
'AAG',
'CUA',
'GGU',
'AGC',
'AGU',
'GGA',
'CCG',
'AAG',
'GCA',
'CCG',
'CUA',
'GUA',
'CGU',
'GGG',
'UCG',
'GAG',
'UAA',
'AGG',
'AAG',
'CUC',
'GGG',

'AUU',
'CCC',
'GUG',
'CUU',
'CCG',
'GUA',
'AAA',
'ACU',
'CUG',
'GUG',
'AGA',
'AAG',
'GCG',
'GAC',
'CUA',
'GAU',
'CUA',
'GAG',
'GGC',
'AAG',
'GAC',
'UGG',
'UAC',
'CGC',
'GGA',
'GGG',
'GAC',
'GUG',
'UAG',
'UAC',
'GUA',
'UUG',
'AGC',
'GAC',
'GUA',
'AAA',
'UCG',
'GGA',
'GGC',
'AUU',
'UUG',
'GUC',
'AAA',
'AUU',
'AUG',
'AUA',
'UAA',
'CCA',
'UUU',
'GAA',
'UCU',
'CGG',
'AAU',
'AGU',
'GCU',
'GUG',
'GCC',
'AAU',
'GGG',
'AGG',
'AGC',
'GAA',
'GGC',
'CGG',
'AUC',

'CGA',
'AUC',
'GCG',
'GAG',
'UAG',
'CUA',
'UGA',
'AGA',
'AUG',
'GGC',
'AUU',
'CUC',
'CAC',
'AGG',
'AUA',
'AGA',
'UCU',
'GGG',
'AAA',
'GUU',
'ACA',
'ACA',
'GCG',
'UUU',
'GUG',
'GAG',
'CUG',
'GCA',
'UAG',
'GUA',
'GAG',
'CAA',
'CGC',
'UAG',
'GCA',
'AAA',
'UCA',
'UGG',
'GAG',
'AAU',
'CCG',
'AUA',
'CAA',
'ACU',
'UGA',
'UGG',
'AGG',
'GCC',
'GGU',
'UUC',
'GCU',
'ACG',
'UCA',
'GAC',
'GAA',
'GCU',
'UAC',
'CGG',
'UGU',
'AGG',
'GGA',
'GAC',
'CAA',
'GUU',
'AAA',

'GAA',
'GAA',
'CGC',
'AUA',
'GAA',
'AAA',
'GUA',
'AGU',
'ACA',
'GAU',
'GCA',
'CGA',
'AAG',
'AUA',
'AUA',
'GUU',
'UAG',
'GAU',
'GAA',
'CGU',
'CCU',
'GAG',
'ACA',
'GGG',
'GGU',
'GCC',
'CAA',
'UGA',
'CUG',
'AGG',
'AUU',
'CGA',
'AGU',
'CGC',
'UGA',
'AAG',
'CAA',
'GUG',
'GGC',
'AUG',
'CAU',
'AGU',
'ACU',
'GAA',
'GGU',
'GUC',
'AAA',
'UUG',
'GCA',
'ACC',
'ACU',
'UGU',
'GGA',
'AGA',
'AUA',
'AAU',
'GAG',
'AAU',
'UGC',
'UCG',
'AGA',
'AUA',
'ACA',
'UGU',
'CGU',

'AGA',
'ACU',
'AAC',
'GAU',
'UUA',
'CGG',
'ACG',
'CAU',
'GGC',
'GAG',
'AGG',
'GCU',
'GGU',
'GAU',
'AGC',
'GAC',
'GAU',
'GGA',
'UUA',
'AAC',
'CGA',
'UUU',
'AAU',
'AGG',
'GGA',
'ACU',
'UAA',
'GAU',
'UCG',
'GAA',
'GGA',
'GGA',
'AGG',
'AGG',
'GAC',
'AGC',
'AGA',
'ACU',
'CUA',
'AUG',
'AGG',
'UAC',
'GGU',
'AAA',
'AAC',
'CUA',
'AAG',
'GUC',
'AGC',
'CUA',
'AGU',
'GAA',
'UGG',
'GUG',
'GGA',
'GGC',
'UCA',
'GUG',
'CCC',
'AAC',
'GAG',
'AGA',
'AGU',
'AGC',
'AUA',

'GCU',
'CAA',
'ACG',
'AGA',
'GAA',
'ACG',
'UUA',
'GGG',
'UGA',
'CUA',
'UAG',
'GAG',
'CGA',
'AGG',
'GAA',
'UGU',
'ACU',
'CAA',
'GGA',
'CGA',
'CGG',
'AGU',
'UAA',
'AAC',
'GAU',
'GGG',
'AAA',
'AUG',
'AGA',
'CCC',
'AGG',
'GAC',
'GCC',
'CCG',
'GGA',
'AAG',
'GGG',
'AGA',
'GAU',
'UUG',
'CAC',
'AAA',
'UGG',
'AAU',
'AAG',
'AUA',
'UAG',
'AGA',
'GAG',
'AUA',
'AUC',
'GCA',
'UAG',
'AAU',
'GGA',
'UUC',
'AGU',
'GAU',
'AGC',
'GCG',
'AGU',
'AAA',
'GUG',
'AGA',
'AGC',

'CUU',
'GGA',
'GGU',
'AGC',
'CUG',
'ACA',
'CCA',
'AUU',
'CAC',
'UGA',
'AUG',
'AGU',
'GGA',
'GUG',
'GAU',
'GGG',
'AAG',
'UUG',
'GUA',
'UUA',
'AGU',
'ACU',
'AUG',
'CAA',
'AGG',
'UCC',
'GCC',
'AUG',
'CAA',
'AAU',
'AAG',
'CAC',
'CGU',
'UUG',
'CCA',
'GCU',
'UAG',
'ACC',
'AGA',
'UGG',
'UAU',
'GGC',
'UAU',
'CGA',
'AGC',
'ACU',
'CAA',
'AAU',
'CGG',
'GGC',
'UAA',
'CCA',
'CGG',
'CGU',
'GAG',
'GGC',
'AGC',
'UAC',
'CAU',
'GAG',
'GAA',
'GCA',
'CCC',
'UGA',
'GGU',

'AUG',
'GAU',
'CGA',
'GUC',
'AGA',
'GUC',
'ACG',
'GGC',
'GUC',
'GUU',
'GCA',
'GAA',
'AUU',
'UAU',
'ACG',
'UCU',
'AGA',
'UUU',
'UGA',
'GAA',
'UGC',
'GAA',
'GGG',
'GGG',
'GGA',
'AGA',
'AUC',
'UCA',
'UGG',
'AGA',
'CAC',
'AAA',
'UGA',
'GAU',
'AGC',
'AAG',
'UGC',
'UGC',
'CAA',
'GUC',
'UUG',
'AAG',
'GUG',
'GAG',
'GUU',
'AAG',
'AUG',
'GGA',
'UAC',
'UAG',
'UGA',
'GUA',
'GGG',
'ACA',
'GGC',
'GGC',
'GUG',
'AAC',
'AAG',
'CUG',
'UAG',
'GGC',
'UGG',
'UAG',
'UAC',

'AGG',
'UCG',
'GGA',
'GAA',
'GGG',
'CUG',
'ACC',
'UAU',
'UGU',
'GGC',
'GAG',
'GGG',
'UCU',
'UUG',
'CUA',
'AGG',
'AAU',
'GGC',
'UGC',
'AGA',
'GAG',
'GAC',
'GGC',
'CCG',
'CGA',
'CAC',
'GGA',
'GGG',
'AUU',
'AGU',
'ACC',
'GGU',
'CUA',
'GAU',
'UCA',
'GAG',
'AGA',
'GUA',
'UAU',
'UGG',
'GGA',
'AGU',
'GUA',
'GGG',
'ACG',
'ACG',
'GCG',
'CAG',
'UGA',
'AAA',
'GAA',
'GUA',
'AGG',
'UAA',
'CGC',
'ACU',
'GGG',
'AAU',
'ACA',
'GAG',
'AGC',
'GUA',
'UAG',
'GUU',
'UAG',

'GUG',
'AAA',
'GGA',
'AGU',
'GUA',
'UCU',
'GAC',
'UGU',
'UUG',
'UUA',
'GAG',
'UUU',
'AUG',
'AGA',
'AAU',
'GUA',
'UAG',
'UUG',
'UGG',
'UAG',
'CUU',
'UAA',
'ACC',
'AAC',
'GAU',
'GGC',
'AUU',
'AUC',
'UAC',
'CGG',
'UGA',
'ACG',
'GGC',
'GCC',
'AGG',
'GCC',
'UAG',
'UAG',
'UGC',
'UAC',
'AUC',
'CAU',
'UAC',
'AUC',
'UGU',
'UGC',
'AGG',
'UUU',
'CGU',
'AAA',
'GGU',
'AAG',
'UCU',
'GAA',
'AUG',
'UGU',
'CAA',
'GUU',
'AAA',
'ACA',
'CCG',
'GCG',
'AUC',
'GCG',
'GGU',

'AGG',
'AGG',
'CCU',
'ACC',
'AGA',
'AUG',
'UGG',
'GGG',
'AAU',
'AGU',
'CGU',
'GGA',
'CUA',
'GGG',
'AAU',
'CCA',
'CAA',
'GAU',
'GGA',
'GAG',
'AAC',
'GAG',
'AUA',
'AGC',
'UGA',
'ACG',
'GUC',
'UUG',
'CGA',
'GUA',
'GGA',
'AUA',
'GAG',
'AGG',
'CGC',
'GAA',
'AAC',
'GGA',
'AUG',
'CAG',
'CAG',
'GCG',
'AUG',
'AUC',
'UAG',
'UAG',
'AGG',
'GCA',
'GAA',
'CGC',
'UCG',
'GGA',
'GGA',
'AGG',
'GUG',
'CGC',
'CGA',
'UAA',
'CAG',
'CGG',
'UAG',
'AAG',
'UAU',
'AGA',
'GUA',

'GAU',
'GGU',
'AAA',
'CGC',
'AAA',
'GUA',
'AGU',
'CAG',
'GAG',
'UUG',
'AUC',
'GUC',
'AGG',
'UUU',
'AAU',
'CGC',
'UGG',
'AUG',
'AAA',
'CCU',
'GCU',
'CGA',
'GGG',
'GUG',
'GGU',
'UGU',
'CUA',
'GAA',
'CCG',
'AUU',
'CAG',
'UGA',
'CUA',
'GCG',
'AAU',
'CAG',
'AAG',
'UCC',
'AAC',
'GGG',
'UGA',
'AAA',
'GCU',
'CAA',
'GAC',
'AUC',
'AUC',
'AUC',
'GAG',
'GCC',
'CUG',
'UGC',
'UUA',
'AAA',
'GUU',
'UGG',
'AAA',
'GCU',
'UUU',
'UAG',
'GCG',
'AGA',
'AUA',
'GAA',
'GAU',

'AGC',
'AUG',
'GUA',
'AUA',
'CAC',
'UAG',
'AGG',
'GCG',
'AGG',
'GAG',
'UUC',
'AGG',
'CAG',
'UAC',
'AAA',
'UAG',
'UAA',
'GAG',
'AGU',
'CCA',
'AAU',
'CAA',
'AGG',
'GUA',
'ACA',
'AGA',
'GGA',
'UAU',
'UAA',
'AAG',
'UUA',
'GGU',
'UAG',
'AUU',
'UGG',
'AAG',
'GGG',
'UGG',
'UCG',
'CGU',
'GCA',
'GGG',
'UCU',
'AUG',
'CGC',
'GAC',
'GAA',
'AAG',
'AUA',
'UGG',
'GGG',
'AGC',
'AAG',
'CCU',
'AGA',
'CGC',
'UUU',
'ACU',
'GGA',
'UAA',
'GGA',
'UUG',
'GGC',
'UUA',
'UUA',

```
'UCA',
'UGG',
'CAA',
'GUU',
'AAG',
'CAG',
'UGG',
'UAA',
'GCA',
'UUA',
'CAA',
'UUU',
'CGU',
'UAG',
'CAC',
'GGA',
'UAG',
'AGU',
'GUC',
'CGA',
'GCG',
'GCA',
'CGG',
'GAA',
'AGU',
'UUA',
'UGG',
'UGU',
'UUU',
'AAA',
'UAU',
'GAU',
'UUC',
'UGG',
'AAG',
'GAG',
'UAC',
'AAA',
'UGA',
'GGG',
'AAC',
'GUC',
'GUU',
'UGA',
'UGU',
'...']
```

Make a dataframe of anticodons and amino acids.

```
In [338...]: translation_df2=pd.DataFrame(columns=['Anticodons', 'Amino_Acids'])
for i in anticodons:
    translation_df2.loc[len(translation_df2.index)] = [i, i]
translation_df2
```

	Anticodons	Amino_Acids
0	UAC	UAC
1	GAC	GAC
2	AGG	AGG
3	AUA	AUA
4	UAC	UAC

	Anticodons	Amino_Acids
...
9126	AAG	AAG
9127	GCC	GCC
9128	ACA	ACA
9129	AGG	AGG
9130	C	C

9131 rows × 2 columns

Remove extra nucleotide.

```
In [339]: translation_df2=translation_df2.drop(9130)
translation_df2
```

	Anticodons	Amino_Acids
0	UAC	UAC
1	GAC	GAC
2	AGG	AGG
3	AUA	AUA
4	UAC	UAC
...
9125	GGU	GGU
9126	AAG	AAG
9127	GCC	GCC
9128	ACA	ACA
9129	AGG	AGG

9130 rows × 2 columns

Anticodon conversion to amino acid chart.

- UUU Phe
- UCU Ser
- UAU Tyr
- UGU Cys
- UUC Phe
- UCC Ser
- UAC Tyr
- UGC Cys
- UUA Leu
- UCA Ser

- UAA Stop
- UGA Stop
- UUG Leu
- UCG Ser
- UAG Stop
- UGG Trp
- CUU Leu
- CCU Pro
- CAU His
- CGU Arg
- CUC Leu
- CCC Pro
- CAC His
- CGC Arg
- CUA Leu
- CCA Pro
- CAA Gln
- CGA Arg
- CUG Leu
- CCG Pro
- CAG Gln
- CGG Arg
- AUU Ile
- ACU Thr
- AAU Asn
- AGU Ser
- AUC Ile
- ACC Thr
- AAC Asn
- AGC Ser
- AUA Ile
- ACA Thr
- AAA Lys
- AGA Arg
- AUG Met
- ACG Thr
- AAG Lys
- AGG Arg
- GUU Val
- GCU Ala
- GAU Asp
- GGU Gly
- GUC Val
- GCC Ala

- GAC Asp
- GGC Gly
- GUA Val
- GCA Ala
- GAA Glu
- GGA Gly
- GUG Val
- GCG Ala
- GAG Glu
- GGG Gly

Amino acids chart.

- Ala A Alanine
- Arg R Arginine
- Asn N Asparagine
- Asp D Aspartic acid
- Cys C Cysteine
- Gln Q Glutamine
- Glu E Glutamic acid
- Gly G Glycine
- His H Histidine
- Ile I Isoleucine
- Leu L Leucine
- Lys K Lysine
- Met M Methionine
- Phe F Phenylalanine
- Pro P Proline
- Ser S Serine
- Thr T Threonine
- Trp W Tryptophan
- Tyr Y Tyrosine
- Val V Valine
- Asx B Asn or Asp
- Glx Z Gln or Glu
- Xle J Leu or Ile
- Sec U Selenocysteine
- Pyl O Pyrrolysine
- Unk X Unknown

Match anticodons with amino acids.

```
In [340]: translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].replace('UUU','F')
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].replace('UUC','F')
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].replace('UUA','L')
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].replace('UUG','L')
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].replace('CUU','L')
```


Out[340...]

	Anticodons	Amino_Acids
0	UAC	Y
1	GAC	D
2	AGG	R
3	AUA	I
4	UAC	Y
...
9125	GGU	G
9126	AAG	K
9127	GCC	A
9128	ACA	T
9129	AGG	R

9130 rows × 2 columns

In [341...]

```
translation_df2['Amino_Acids_copy']=translation_df2['Amino_Acids']  
translation_df2
```

Out[341...]

	Anticodons	Amino_Acids	Amino_Acids_copy
0	UAC	Y	Y
1	GAC	D	D
2	AGG	R	R
3	AUA	I	I
4	UAC	Y	Y
...
9125	GGU	G	G
9126	AAG	K	K
9127	GCC	A	A
9128	ACA	T	T
9129	AGG	R	R

9130 rows × 3 columns

Data Exploration

In [288...]

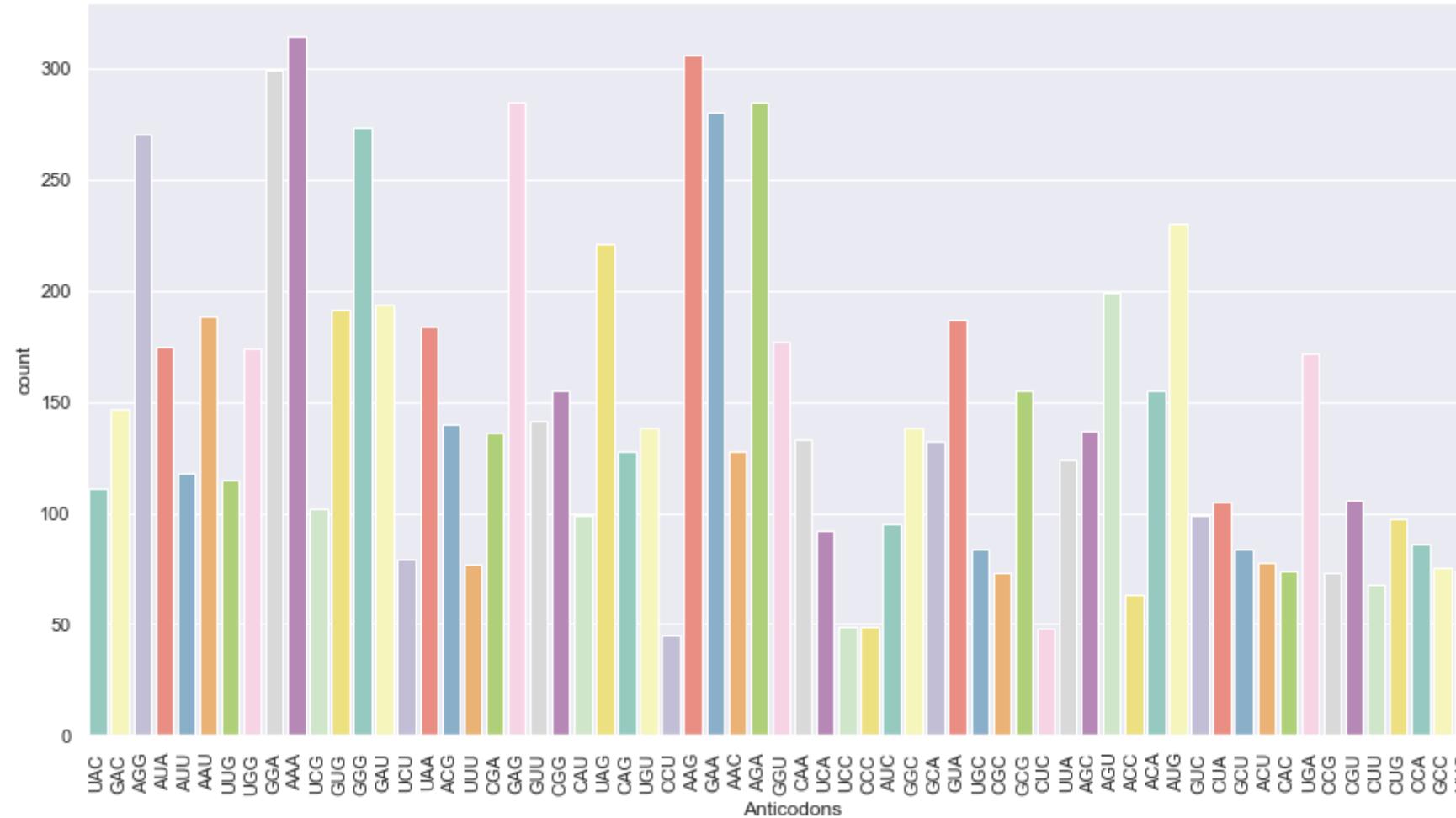
```
plt.figure(figsize=(15,8))  
sns.countplot(translation_df2['Anticodons'], palette='Set3')  
plt.xticks(rotation = 90)
```

Out[288...]

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,  
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
```

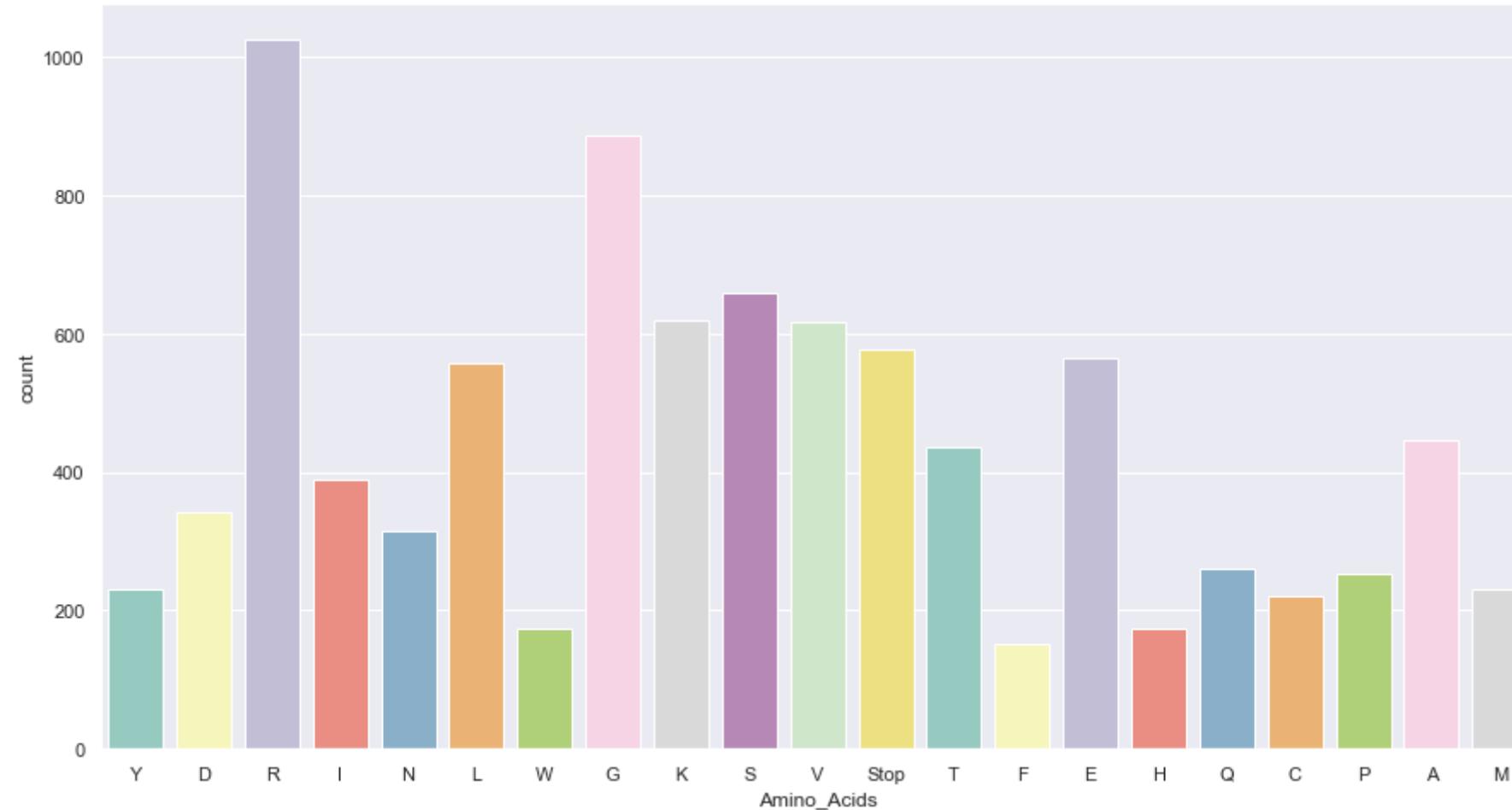
34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63]),
[Text(0, 0, 'UAC'),
Text(1, 0, 'GAC'),
Text(2, 0, 'AGG'),
Text(3, 0, 'AUA'),
Text(4, 0, 'AUU'),
Text(5, 0, 'AAU'),
Text(6, 0, 'UUG'),
Text(7, 0, 'UGG'),
Text(8, 0, 'GGA'),
Text(9, 0, 'AAA'),
Text(10, 0, 'UCG'),
Text(11, 0, 'GUG'),
Text(12, 0, 'GGG'),
Text(13, 0, 'GAU'),
Text(14, 0, 'UCU'),
Text(15, 0, 'UAA'),
Text(16, 0, 'ACG'),
Text(17, 0, 'UUU'),
Text(18, 0, 'CGA'),
Text(19, 0, 'GAG'),
Text(20, 0, 'GUU'),
Text(21, 0, 'CGG'),
Text(22, 0, 'CAU'),
Text(23, 0, 'UAG'),
Text(24, 0, 'CAG'),
Text(25, 0, 'UGU'),
Text(26, 0, 'CCU'),
Text(27, 0, 'AAG'),
Text(28, 0, 'GAA'),
Text(29, 0, 'AAC'),
Text(30, 0, 'AGA'),
Text(31, 0, 'GGU'),
Text(32, 0, 'CAA'),
Text(33, 0, 'UCA'),
Text(34, 0, 'UCC'),
Text(35, 0, 'CCC'),
Text(36, 0, 'AUC'),
Text(37, 0, 'GGC'),
Text(38, 0, 'GCA'),
Text(39, 0, 'GUA'),
Text(40, 0, 'UGC'),
Text(41, 0, 'CGC'),
Text(42, 0, 'GCG'),
Text(43, 0, 'CUC'),
Text(44, 0, 'UUA'),
Text(45, 0, 'AGC'),
Text(46, 0, 'AGU'),
Text(47, 0, 'ACC'),
Text(48, 0, 'ACA'),
Text(49, 0, 'AUG'),
Text(50, 0, 'GUC'),
Text(51, 0, 'CUA'),
Text(52, 0, 'GCU'),
Text(53, 0, 'ACU'),
Text(54, 0, 'CAC'),
Text(55, 0, 'UGA'),
Text(56, 0, 'CCG'),
Text(57, 0, 'CGU'),
Text(58, 0, 'CUU'),
Text(59, 0, 'CUG'),
Text(60, 0, 'CCA'),
Text(61, 0, 'GCC'),

```
Text(62, 0, 'UUC'),  
Text(63, 0, 'UAU'))])
```



```
In [290...]: plt.figure(figsize=(15,8))
sns.countplot(translation_df2[ 'Amino_Acids' ], palette='Set3' )
```

```
Out[290... <AxesSubplot:xlabel='Amino_Acids', ylabel='count'>
```



There is a class imbalance that will be resolved with oversampling.

Feature Engineering

Represent anticodons and amino acids with numbers.

```
In [342...]: translation_df2['Anticodons']=translation_df2['Anticodons'].astype('category')
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].astype('category')
translation_df2['Anticodons']=translation_df2['Anticodons'].cat.codes
translation_df2['Amino_Acids']=translation_df2['Amino_Acids'].cat.codes
```

Reshape X and y so that they are 2d.

```
In [343...]: x2=translation_df2['Anticodons'].values.reshape(-1, 1)
y2=translation_df2['Amino_Acids'].values.reshape(-1, 1)
```

Resample independent and dependent variable.

```
In [344...]: x2_resampled, y2_resampled = oversample.fit_resample(x2, y2)
```

Make a train test split of 80/20.

```
In [345...]: x2_train, x2_test, y2_train, y2_test = train_test_split(x2_resampled, y2_resampled, test_size=0.20, random_state=10)
```

Machine Learning Modeling

Fit the model to the training data.

```
In [50]: decision_tree.fit(X2_train, y2_train)
```

```
Out[50]: DecisionTreeClassifier()
```

Make predictions.

```
In [131]: y2_hat_train_dt = decision_tree.predict(X2_train)
```

All predictions were classified correctly.

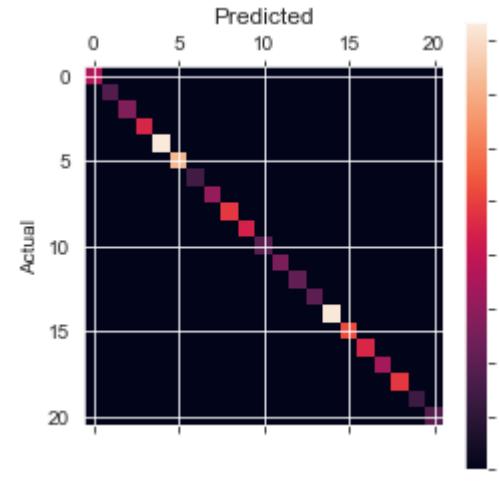
```
In [52]: con_mat(y2_train,y2_hat_train_dt)
```

Confusion Matrix

	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	\
0	370	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
1	0	182	0	0	0	0	0	0	0	0	...	0	0	0	0	
2	0	0	273	0	0	0	0	0	0	0	...	0	0	0	0	
3	0	0	0	447	0	0	0	0	0	0	...	0	0	0	0	
4	0	0	0	0	833	0	0	0	0	0	...	0	0	0	0	
5	0	0	0	0	0	720	0	0	0	0	...	0	0	0	0	
6	0	0	0	0	0	0	142	0	0	0	...	0	0	0	0	
7	0	0	0	0	0	0	0	315	0	0	...	0	0	0	0	
8	0	0	0	0	0	0	0	0	485	0	...	0	0	0	0	
9	0	0	0	0	0	0	0	0	0	437	...	0	0	0	0	
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	...	259	0	0	0	
12	0	0	0	0	0	0	0	0	0	0	...	0	209	0	0	
13	0	0	0	0	0	0	0	0	0	0	...	0	0	200	0	
14	0	0	0	0	0	0	0	0	0	0	...	0	0	0	824	
15	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	

	15	16	17	18	19	20
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	522	0	0	0	0	0
16	0	451	0	0	0	0
17	0	0	339	0	0	0
18	0	0	0	488	0	0
19	0	0	0	0	134	0
20	0	0	0	0	0	183

[21 rows x 21 columns]



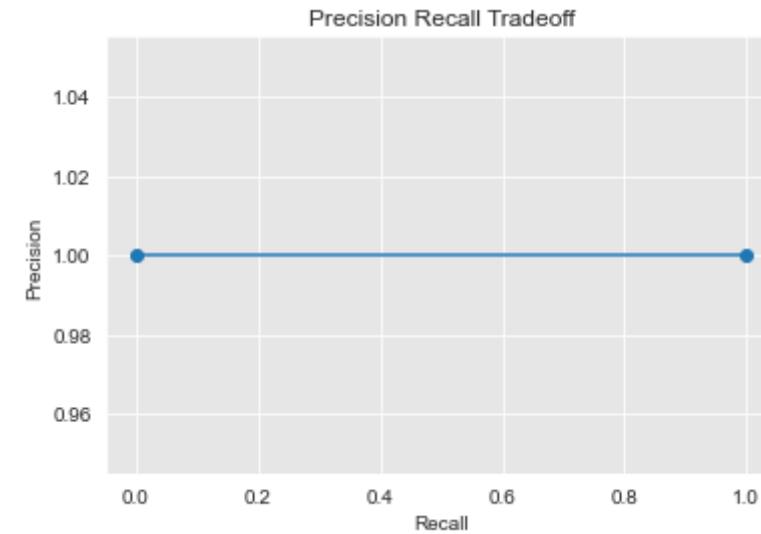
Separate actual and predicted values into each class for metrics.

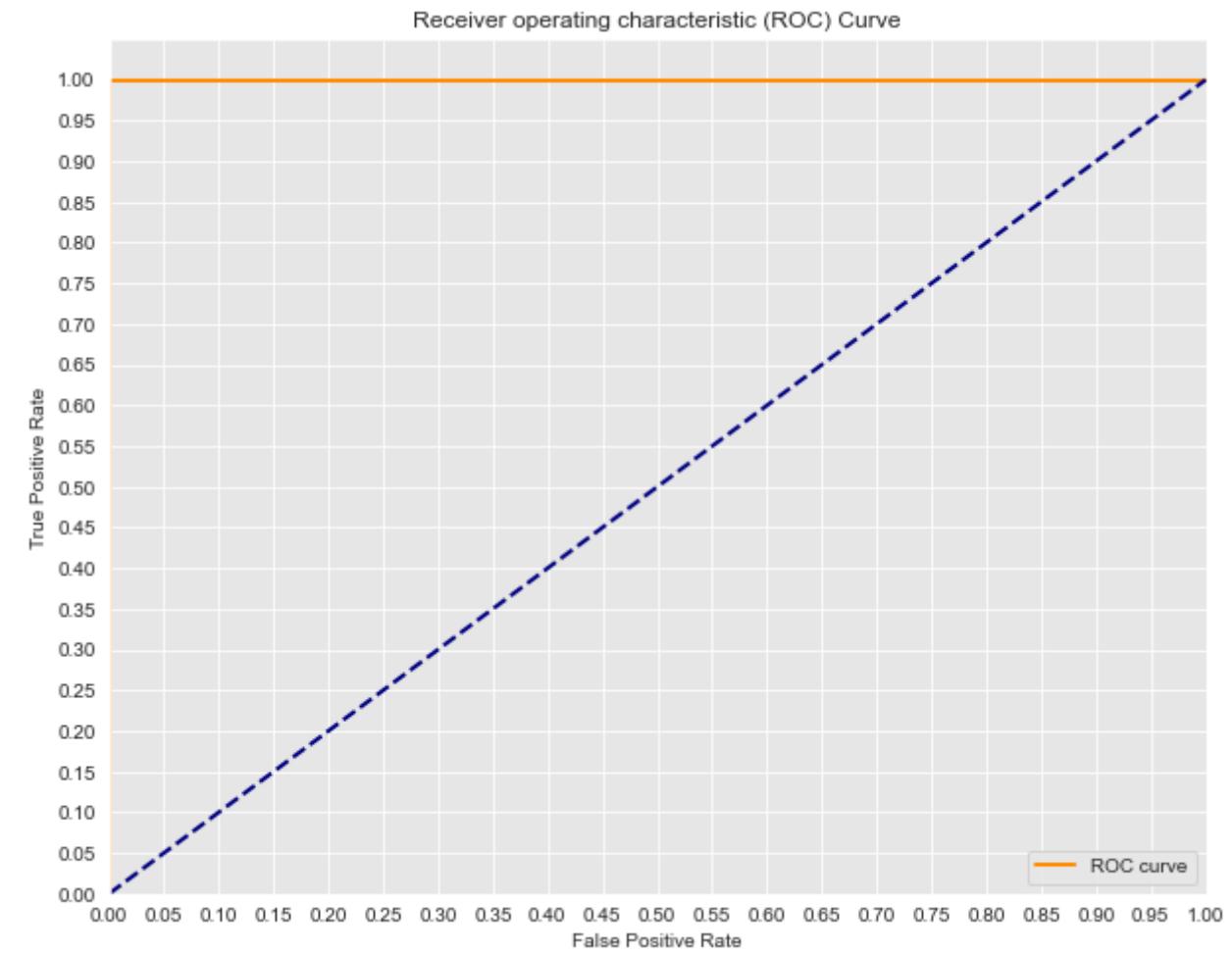
```
In [132]:  
actual_class0=y2_train==0  
actual_class1=y2_train==1  
actual_class2=y2_train==2  
actual_class3=y2_train==3  
actual_class4=y2_train==4  
actual_class5=y2_train==5  
actual_class6=y2_train==6  
actual_class7=y2_train==7  
actual_class8=y2_train==8  
actual_class9=y2_train==9  
actual_class10=y2_train==10  
actual_class11=y2_train==11  
actual_class12=y2_train==12  
actual_class13=y2_train==13  
actual_class14=y2_train==14  
actual_class15=y2_train==15  
actual_class16=y2_train==16  
actual_class17=y2_train==17  
actual_class18=y2_train==18  
actual_class19=y2_train==19  
actual_class20=y2_train==20  
  
predictions_class0=y2_hat_train_dt==0  
predictions_class1=y2_hat_train_dt==1  
predictions_class2=y2_hat_train_dt==2  
predictions_class3=y2_hat_train_dt==3  
predictions_class4=y2_hat_train_dt==4  
predictions_class5=y2_hat_train_dt==5  
predictions_class6=y2_hat_train_dt==6  
predictions_class7=y2_hat_train_dt==7  
predictions_class8=y2_hat_train_dt==8  
predictions_class9=y2_hat_train_dt==9  
predictions_class10=y2_hat_train_dt==10  
predictions_class11=y2_hat_train_dt==11  
predictions_class12=y2_hat_train_dt==12  
predictions_class13=y2_hat_train_dt==13  
predictions_class14=y2_hat_train_dt==14  
predictions_class15=y2_hat_train_dt==15  
predictions_class16=y2_hat_train_dt==16  
predictions_class17=y2_hat_train_dt==17  
predictions_class18=y2_hat_train_dt==18  
predictions_class19=y2_hat_train_dt==19  
predictions_class20=y2_hat_train_dt==20
```

```
a=[actual_class0,actual_class1,actual_class2,actual_class3,actual_class4,actual_class5,actual_class6,actual_class7,  
actual_class8,actual_class9,actual_class10,actual_class11,actual_class12,actual_class13,actual_class14,actual_class15  
,actual_class16,actual_class17,actual_class18,actual_class19,actual_class20]  
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3,predictions_class4,predictions_class5,predictions_class6  
,predictions_class7,predictions_class8,predictions_class9,predictions_class10,predictions_class11,predictions_class12  
,predictions_class13,predictions_class14,predictions_class15,predictions_class16,predictions_class17,predictions_class18  
,predictions_class19,predictions_class20]
```

```
In [61]: multiclass(decision_tree,X2_train,a,p,translation_df2[ 'Amino_Acids_copy' ])
```

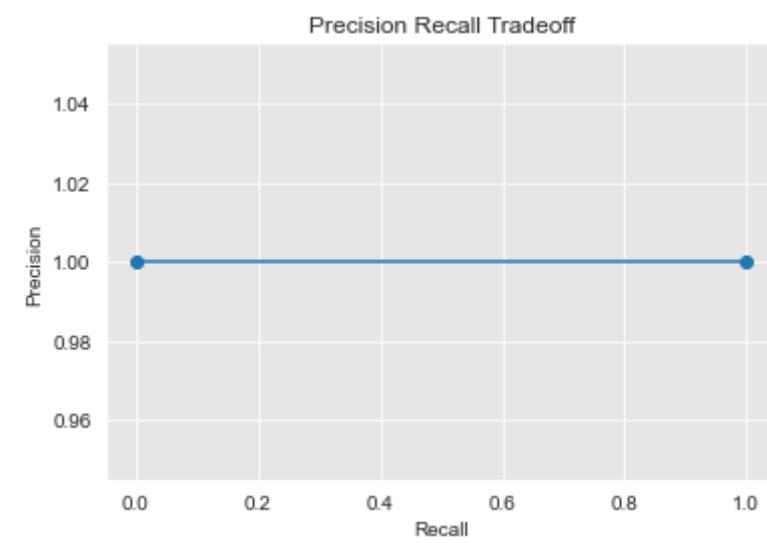
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

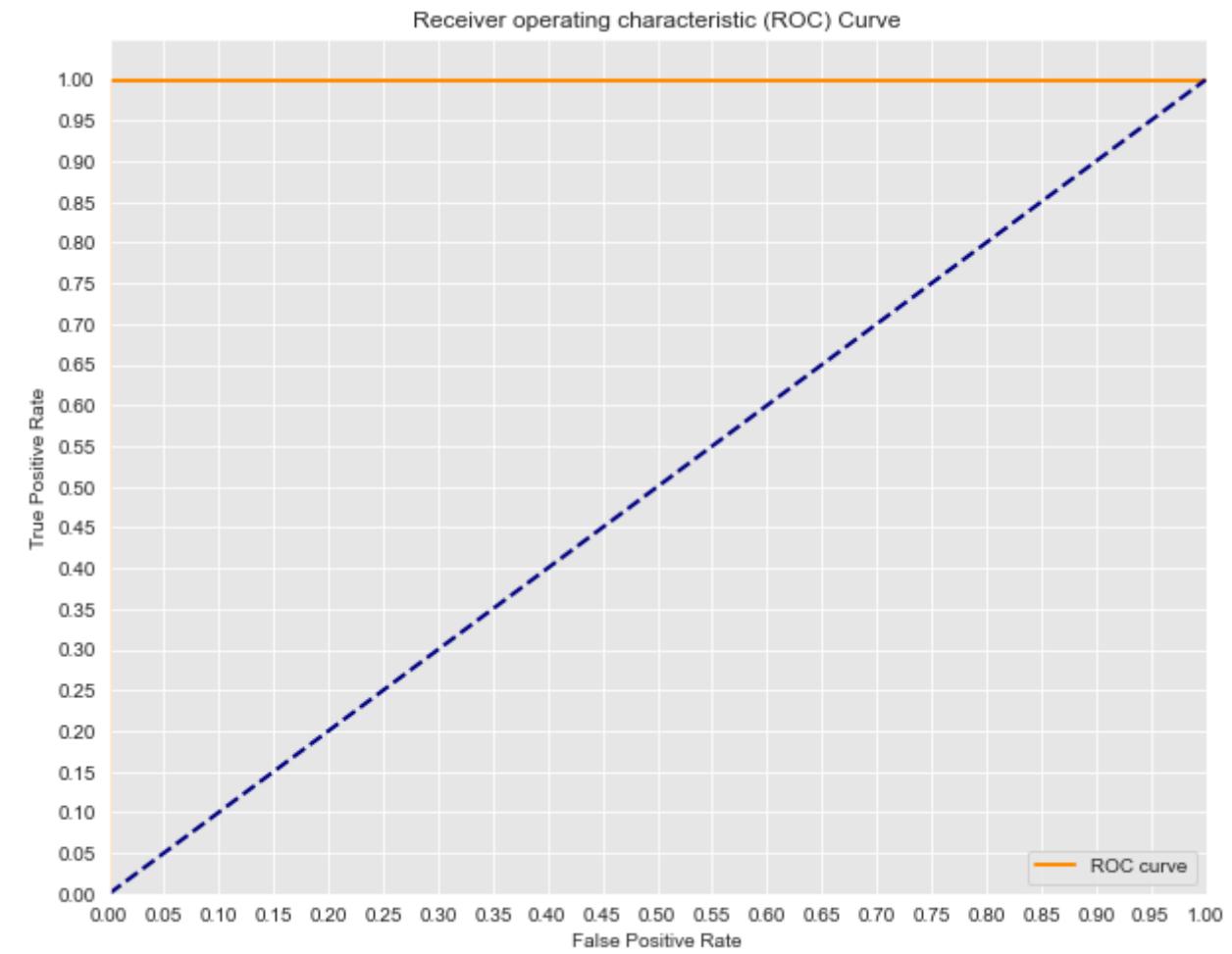




Cross Validated ROC AUC score: 1.0

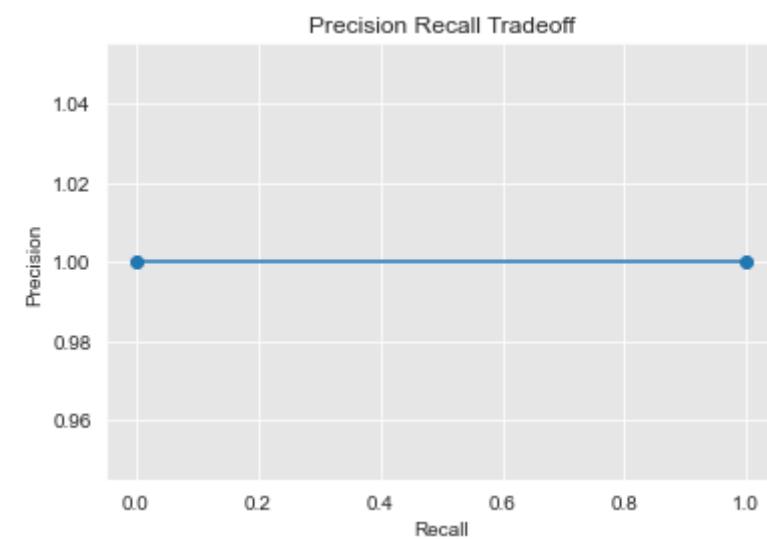
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

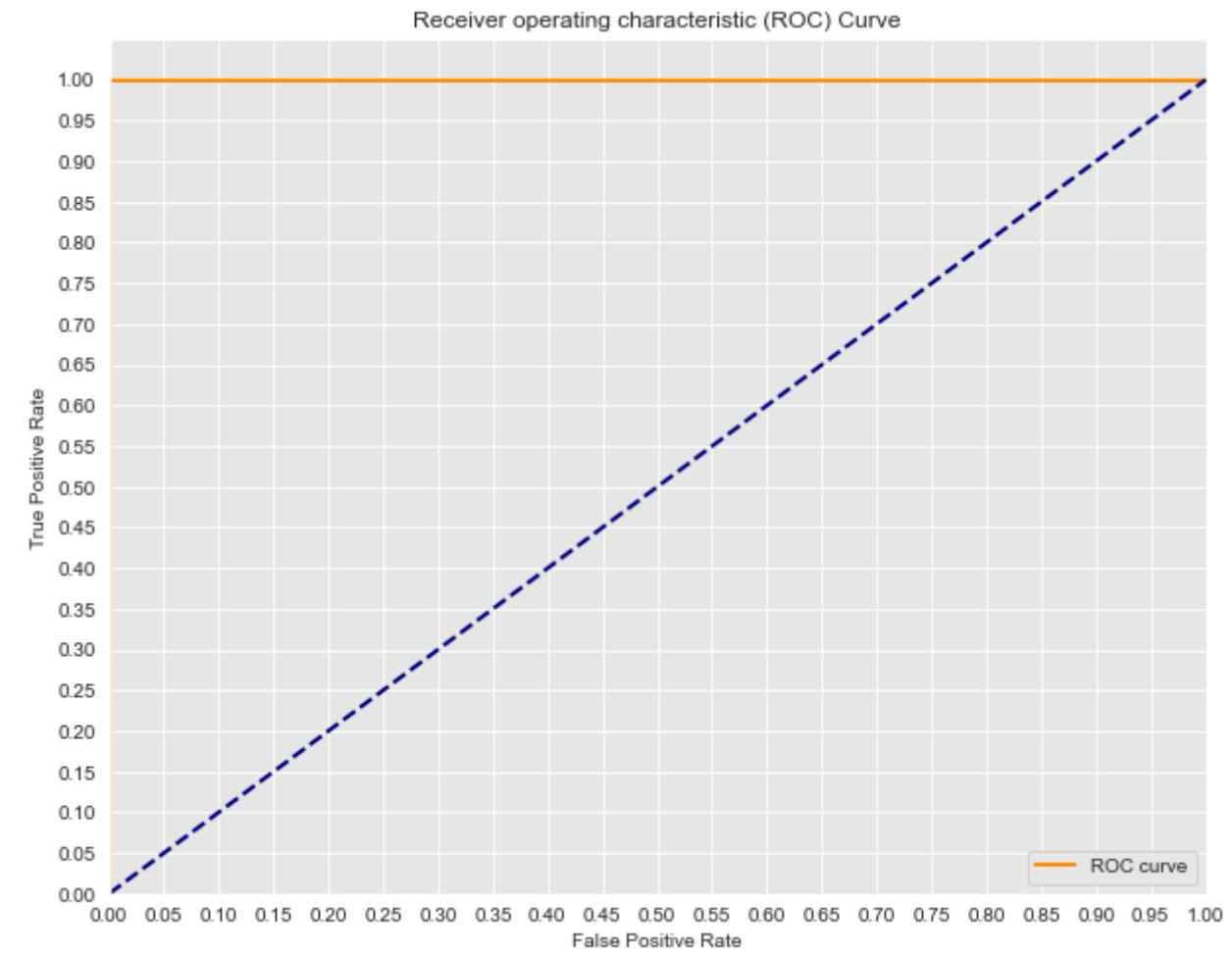




Cross Validated ROC AUC score: 1.0

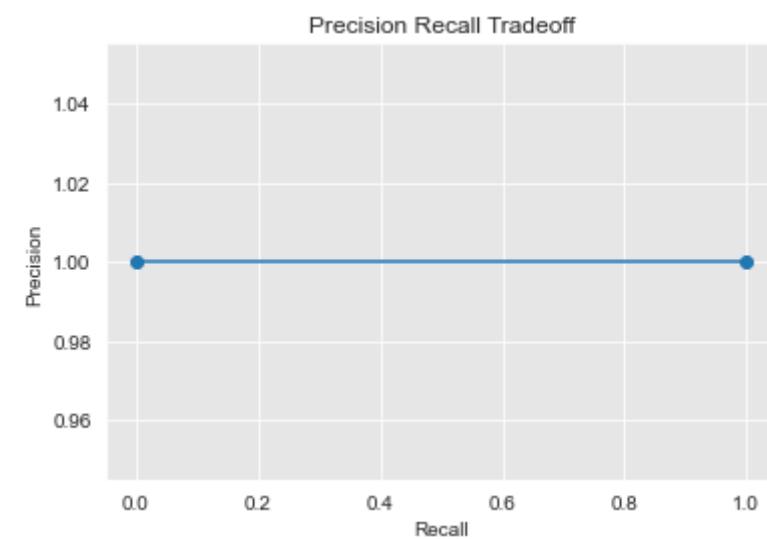
Class:D
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

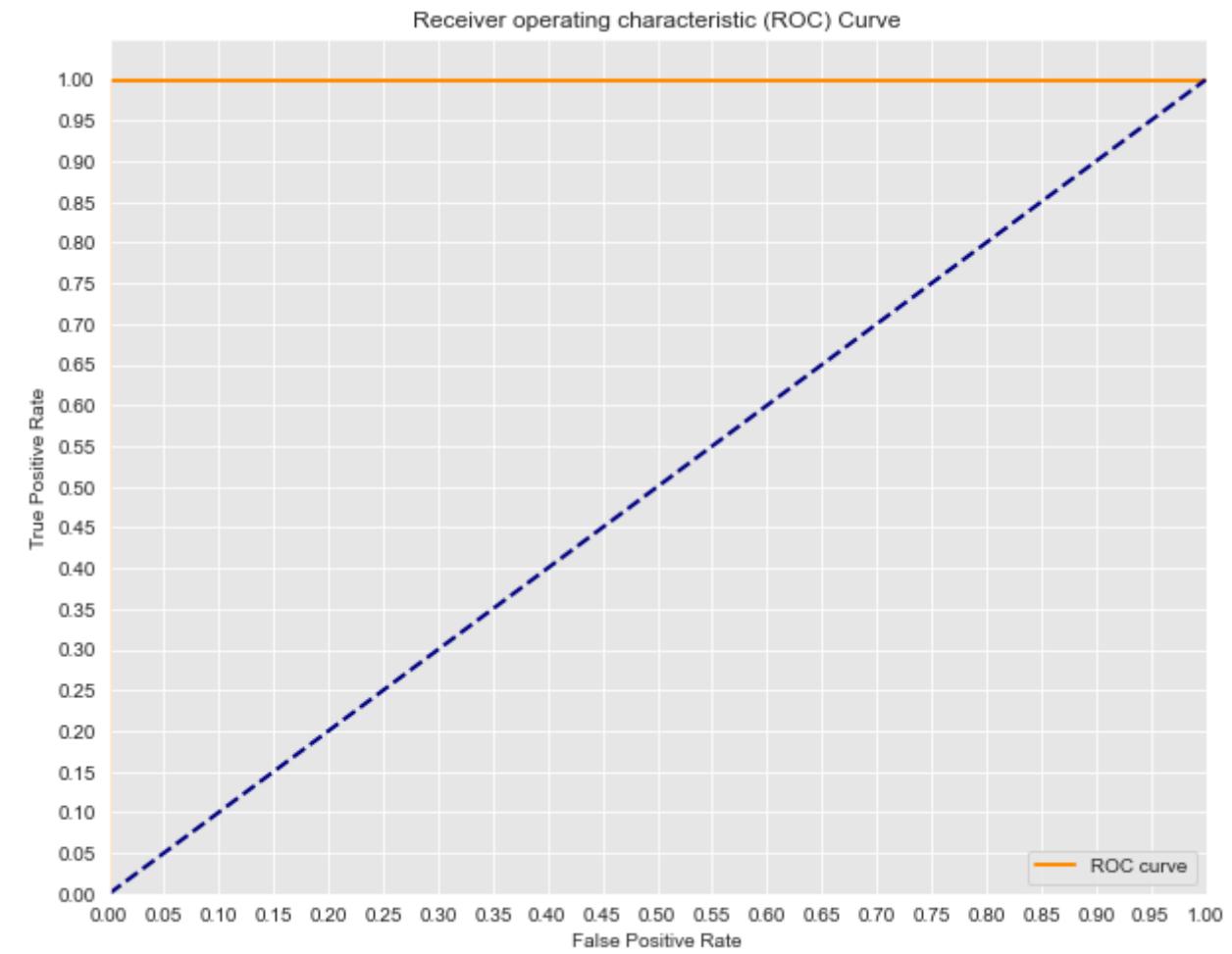




Cross Validated ROC AUC score: 1.0

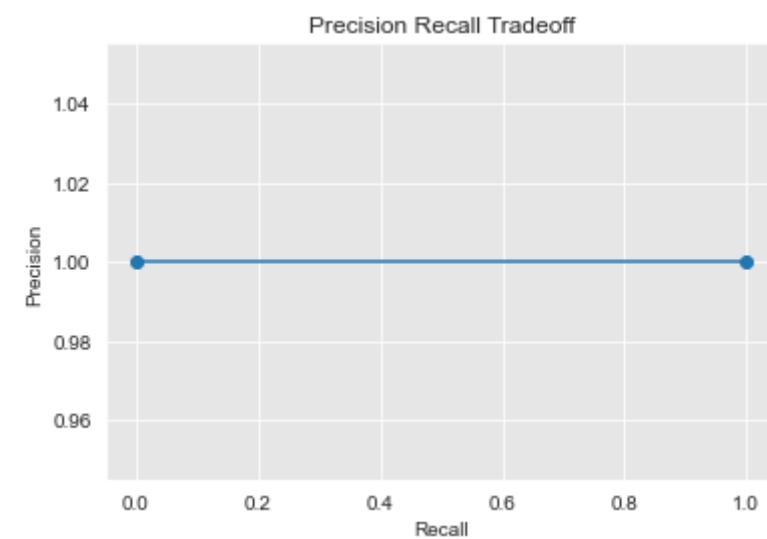
Class:E
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

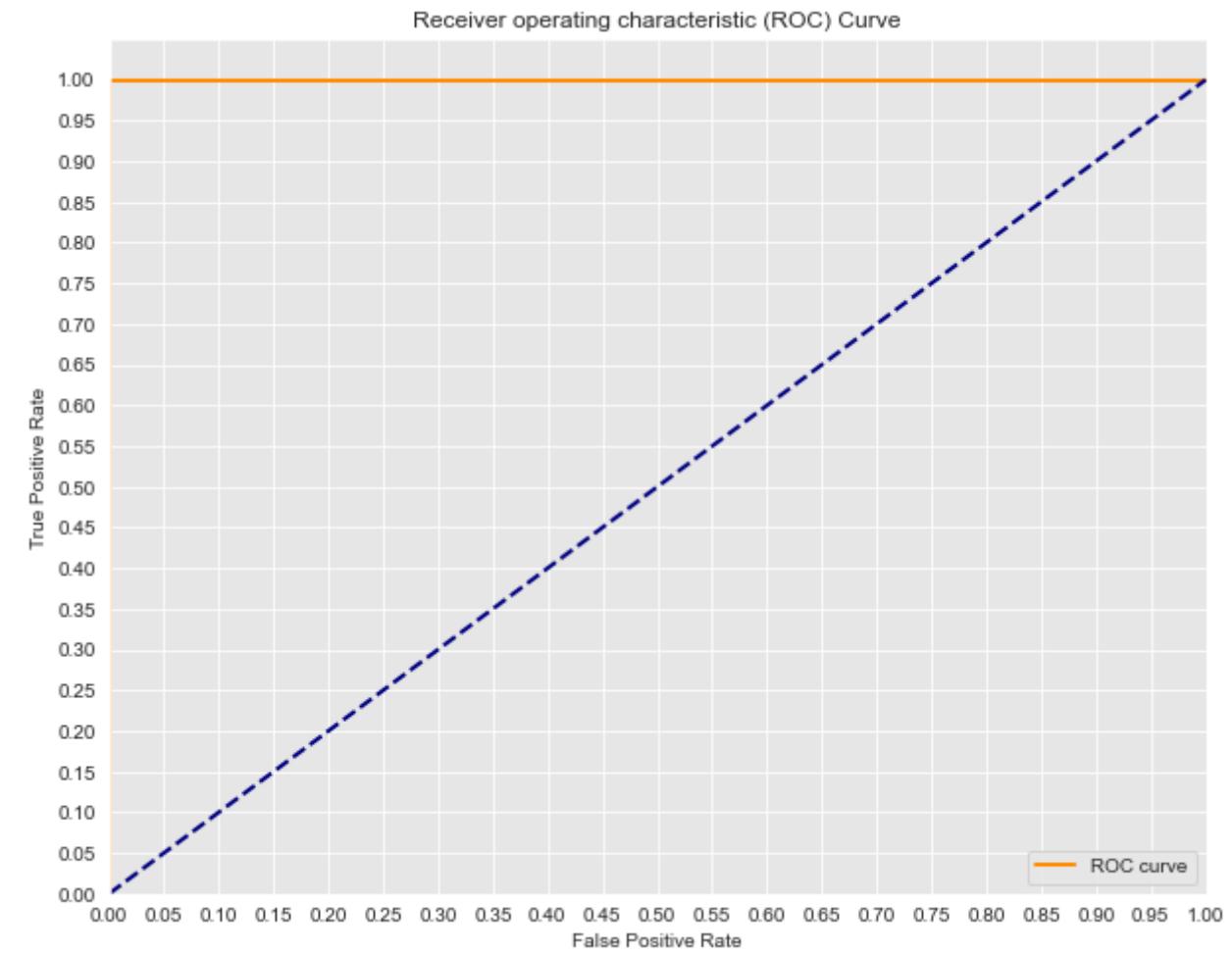




Cross Validated ROC AUC score: 1.0

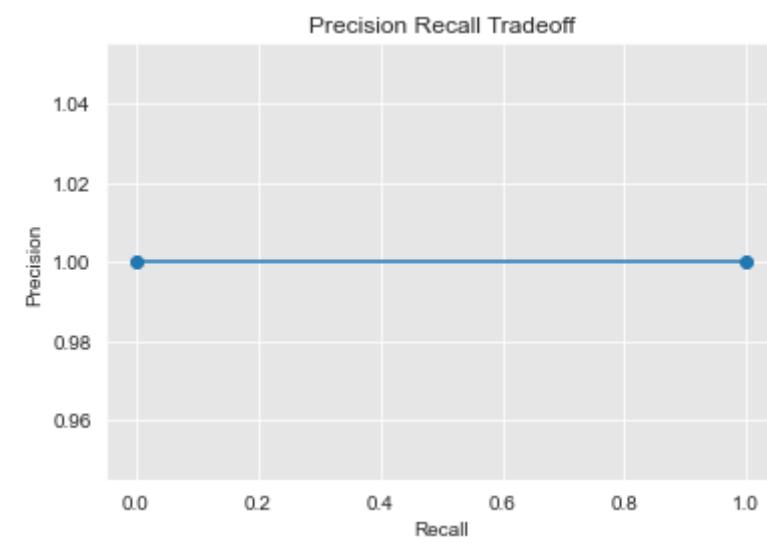
Class:F
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

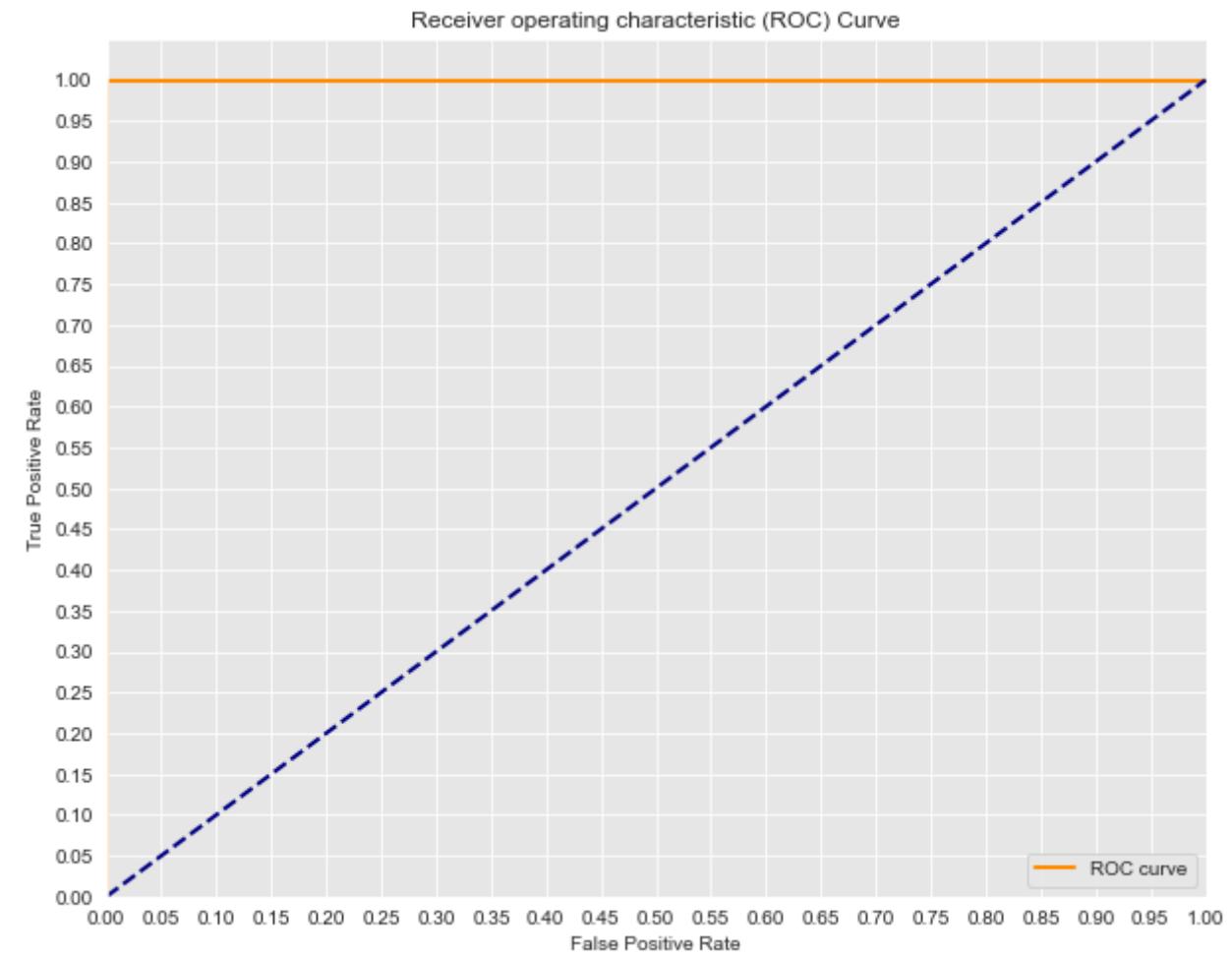




Cross Validated ROC AUC score: 1.0

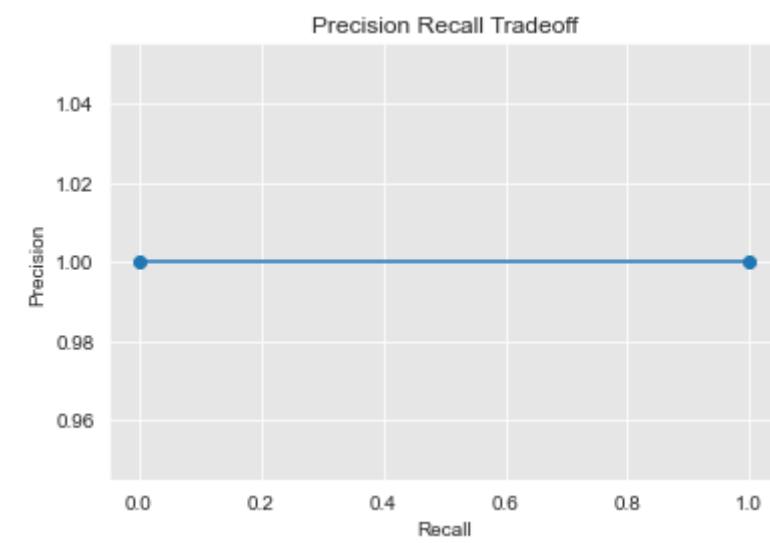
Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

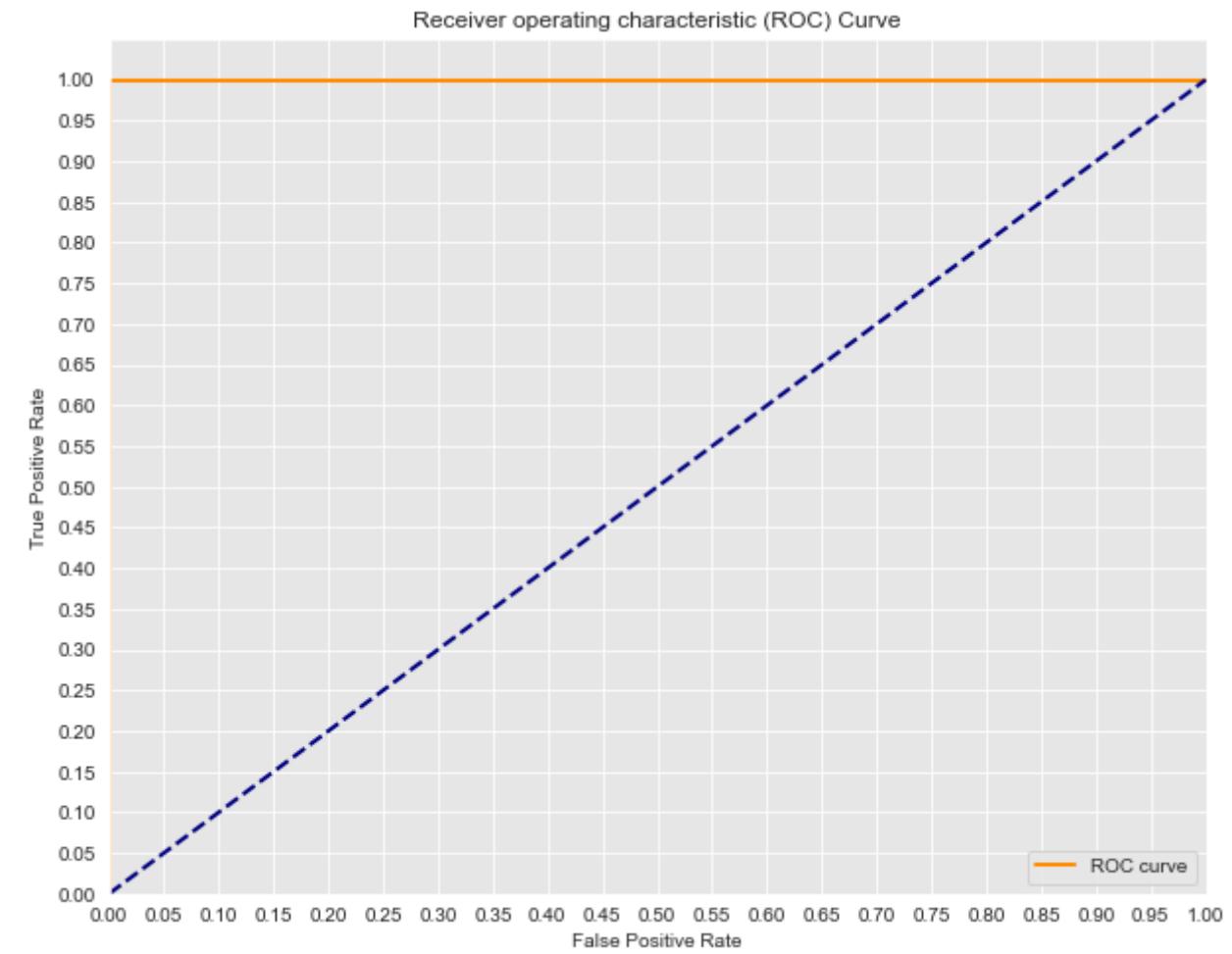




Cross Validated ROC AUC score: 1.0

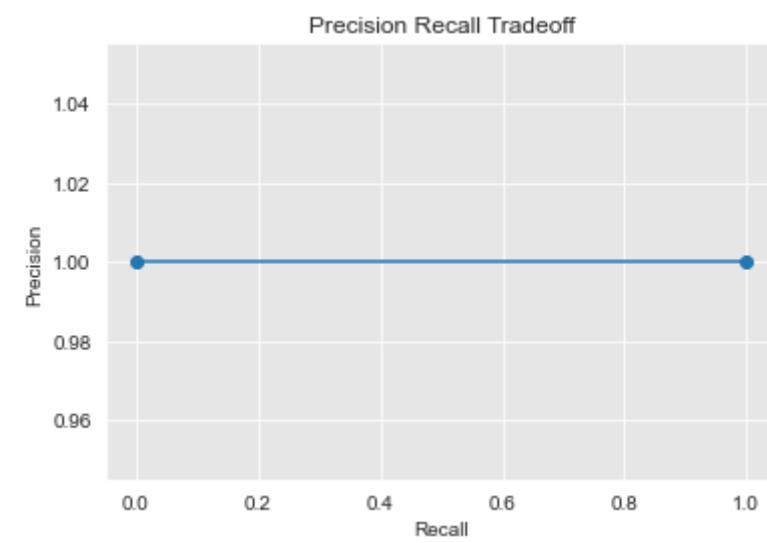
Class:H
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

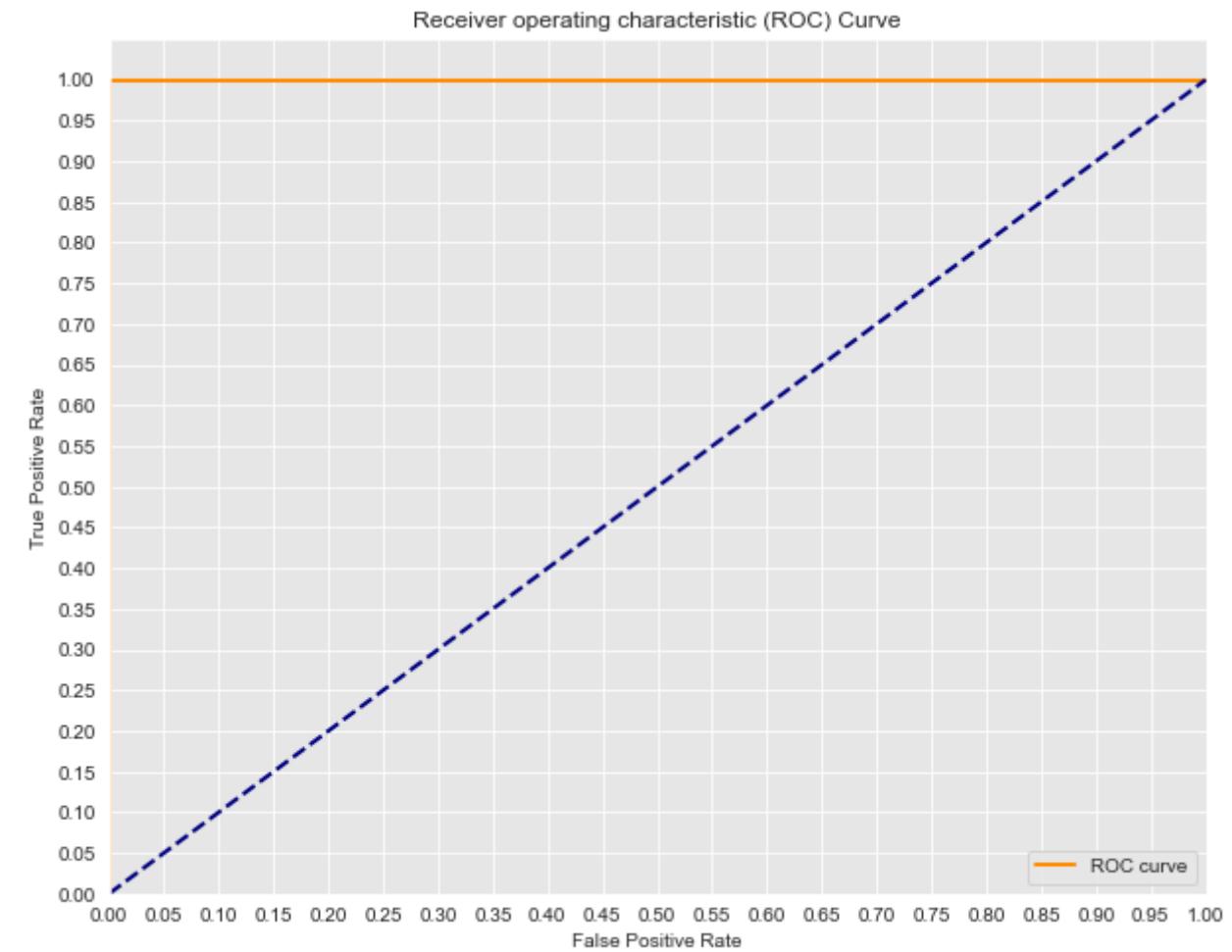




Cross Validated ROC AUC score: 1.0

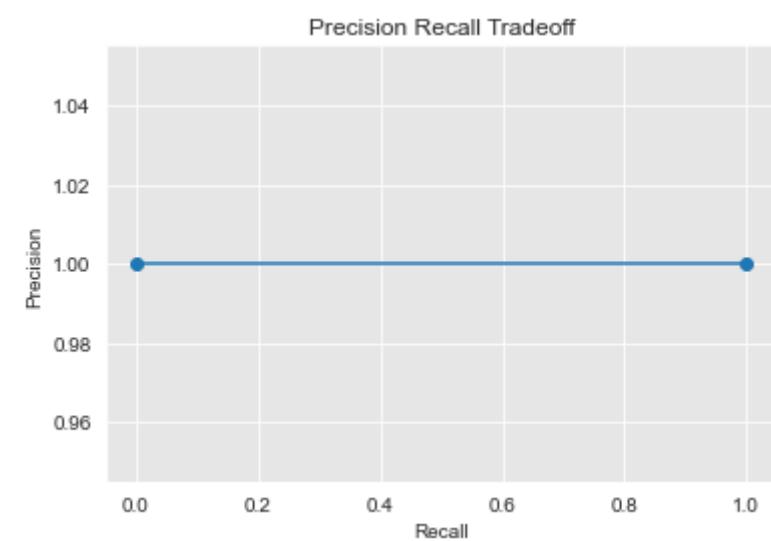
Class:I
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

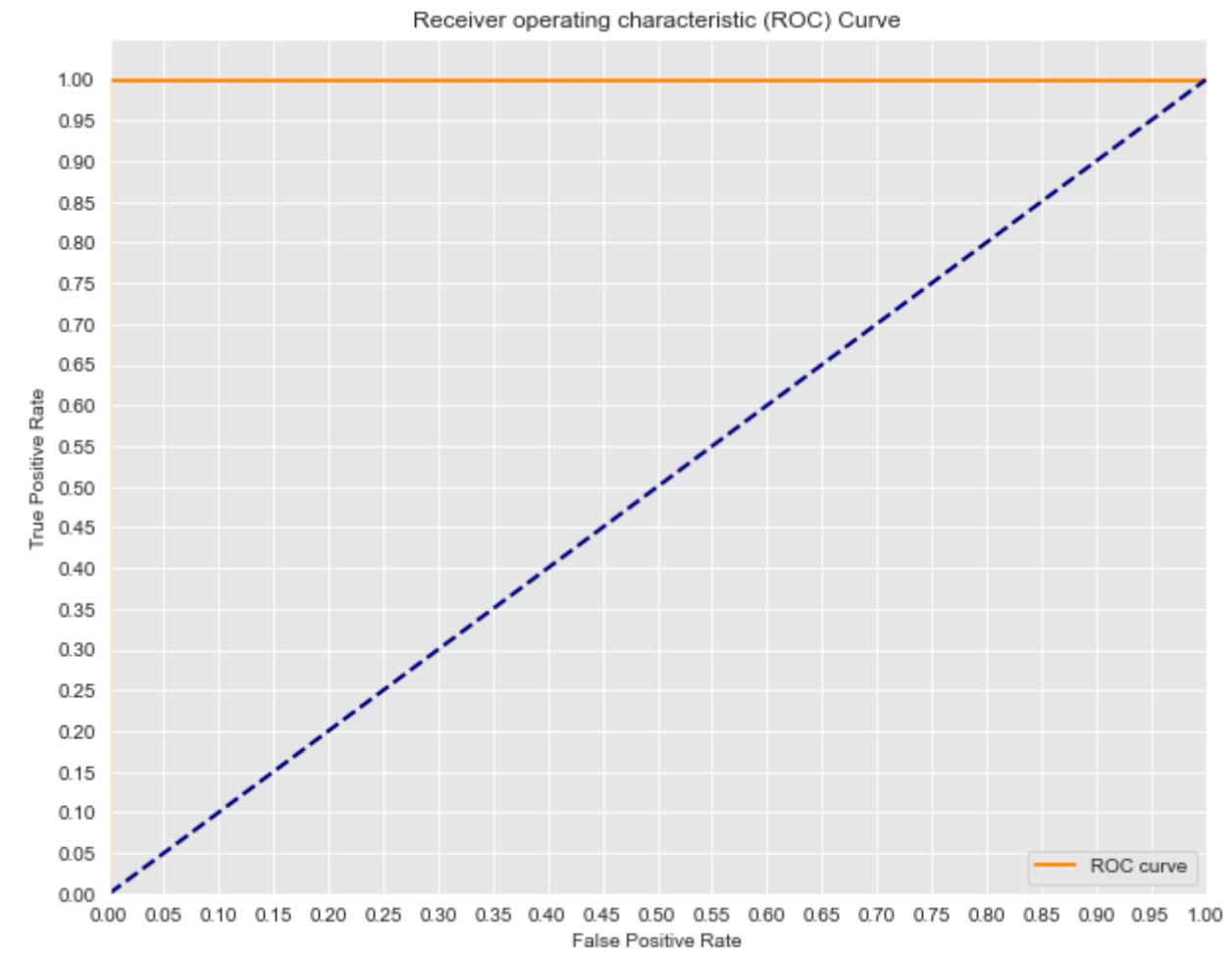




Cross Validated ROC AUC score: 1.0

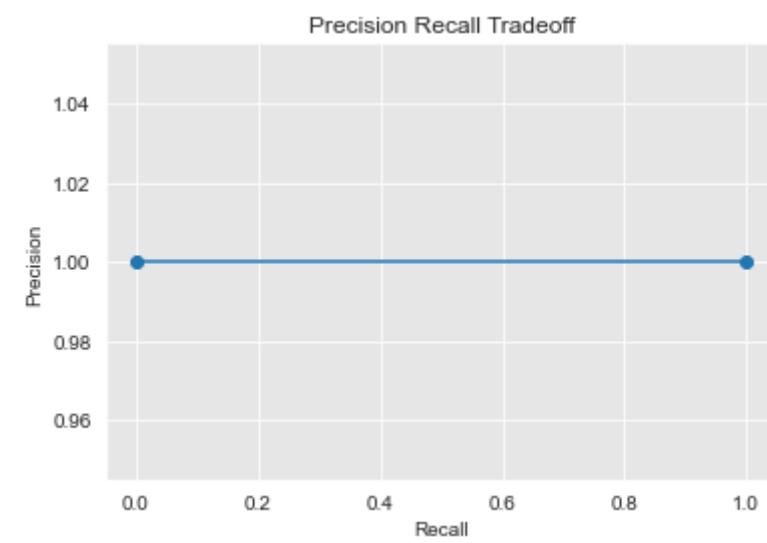
Class:K
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

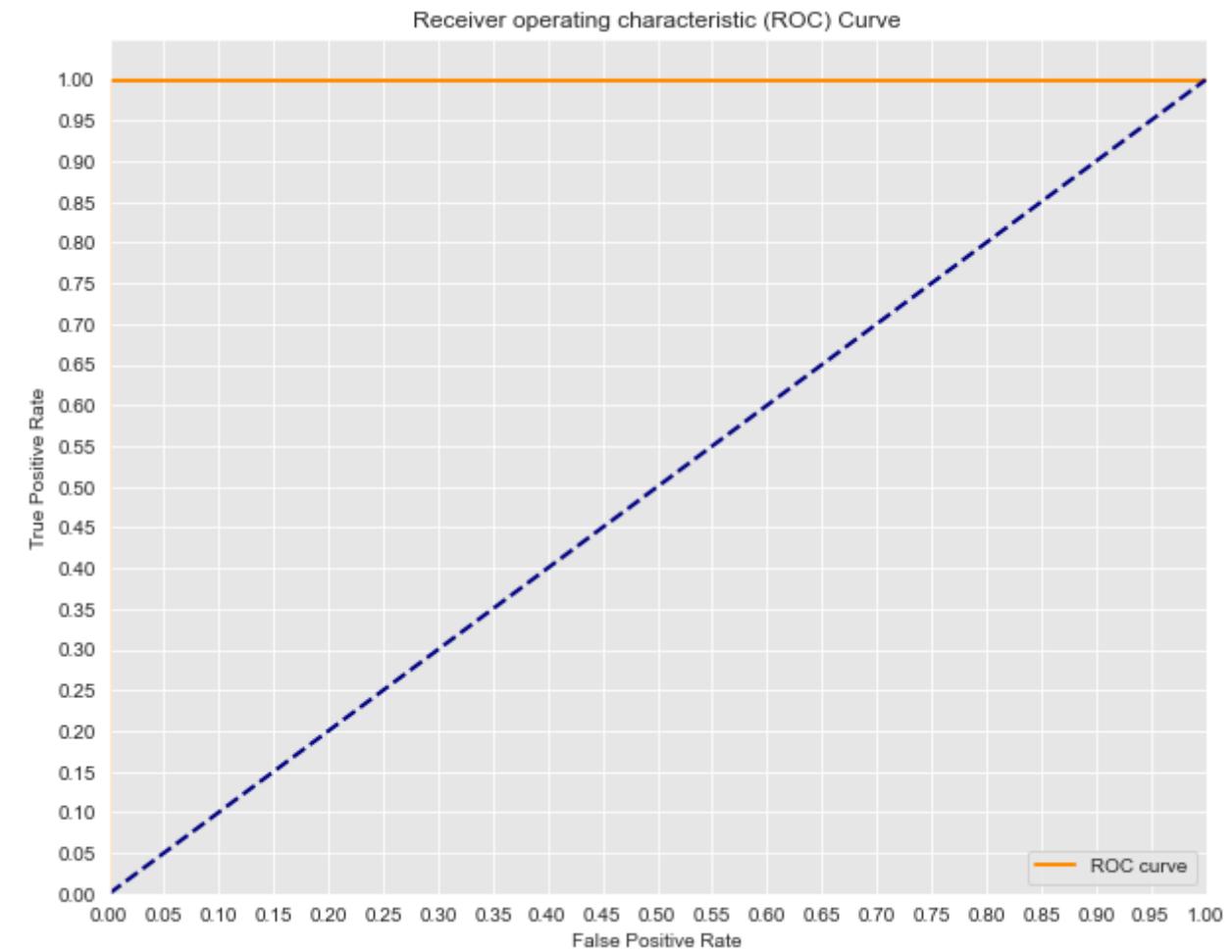




Cross Validated ROC AUC score: 1.0

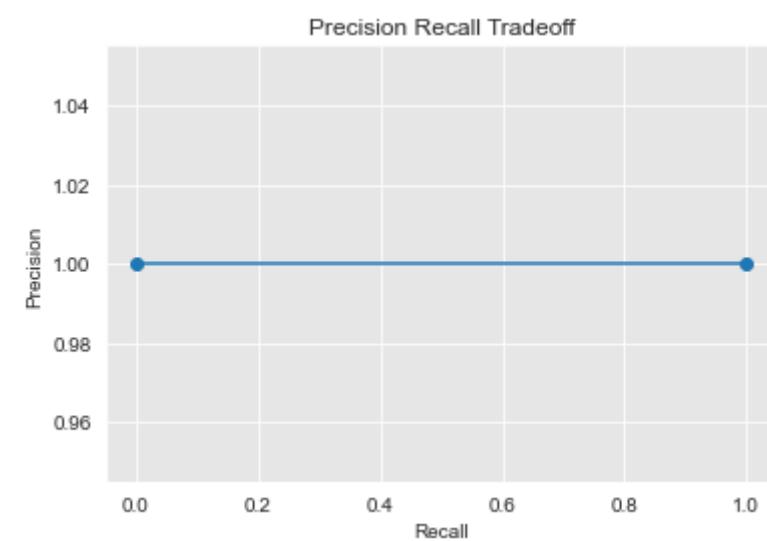
Class:L
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

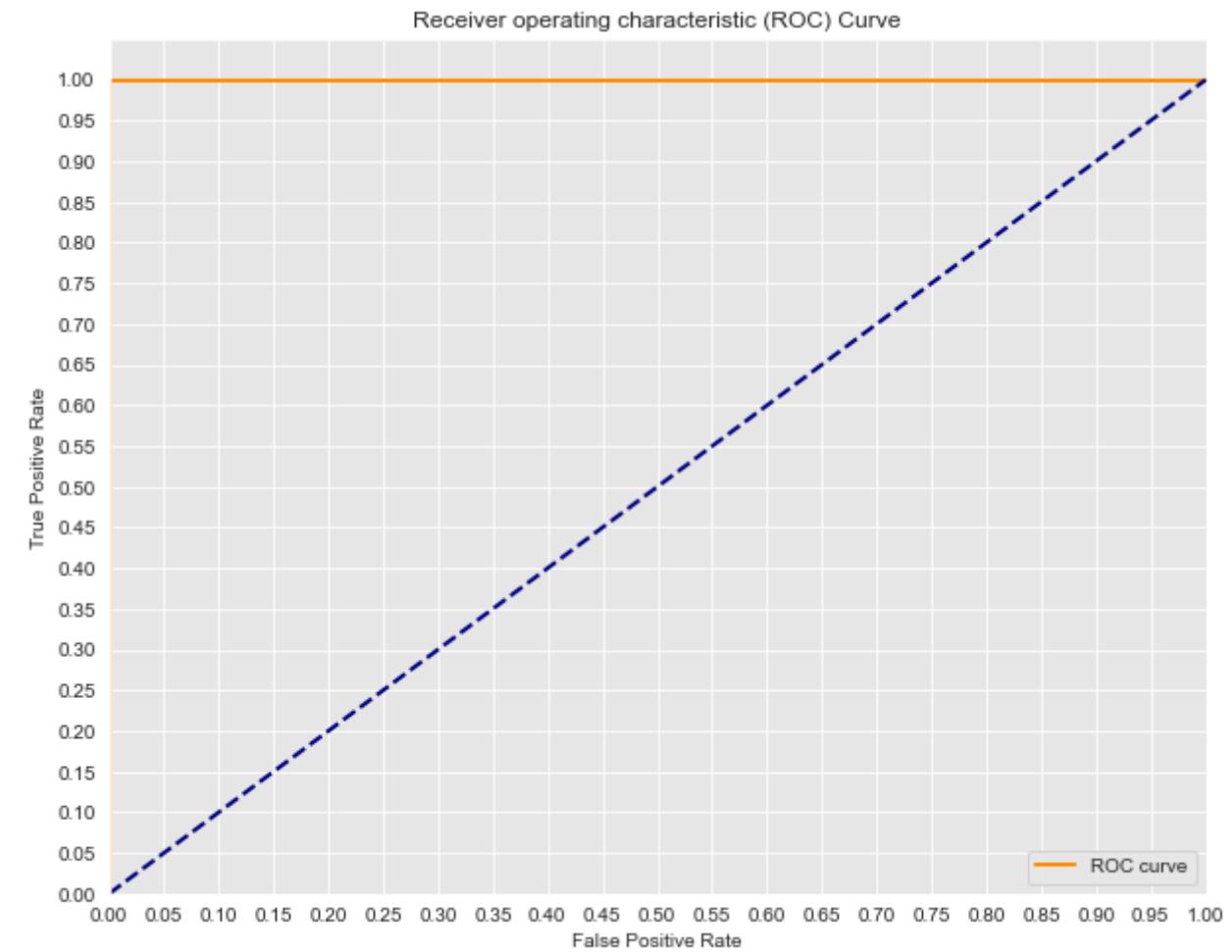




Cross Validated ROC AUC score: 1.0

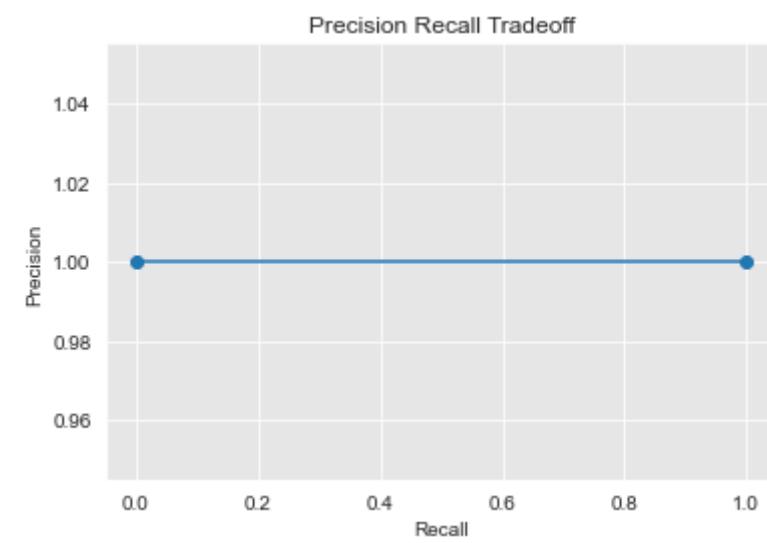
Class:M
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

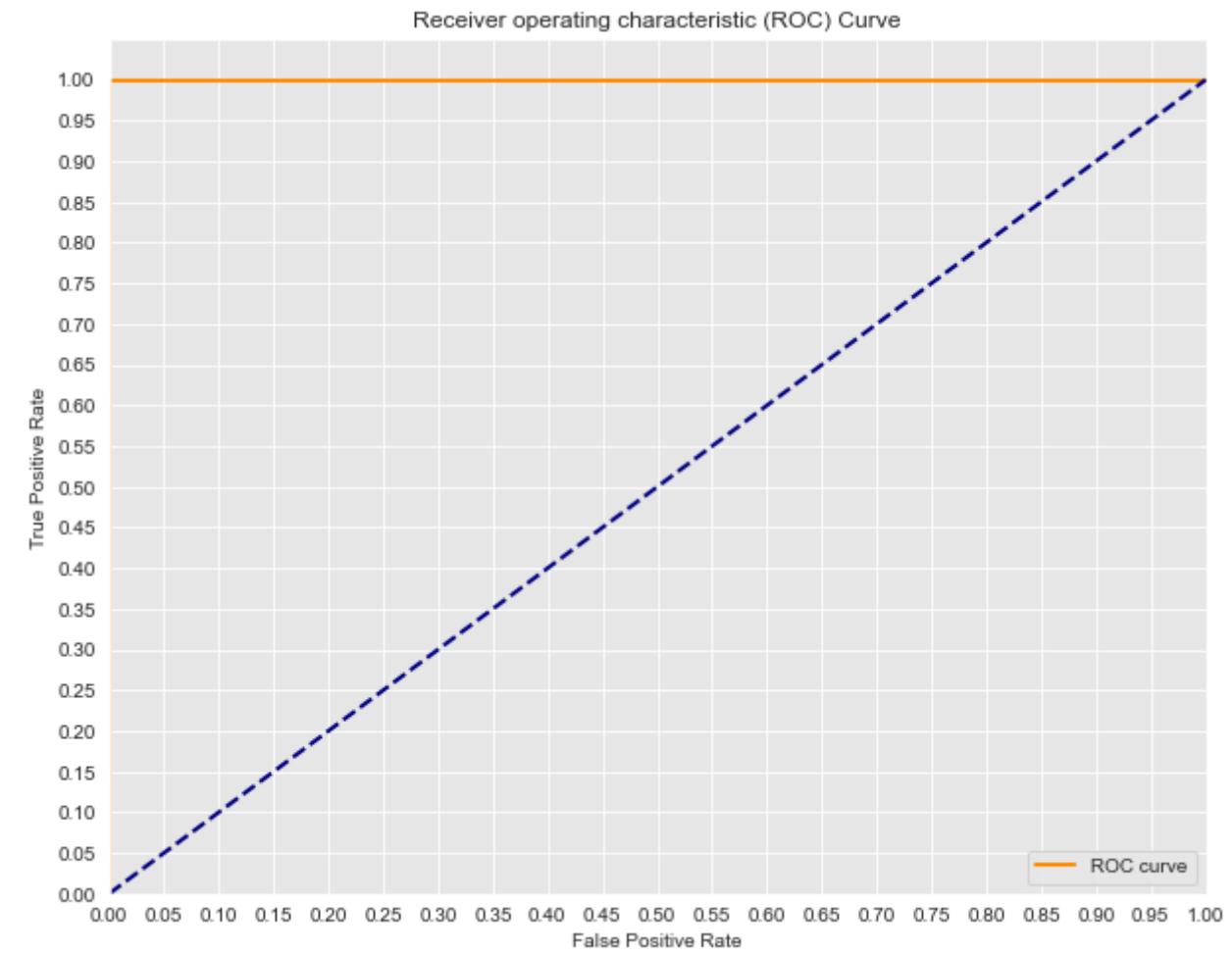




Cross Validated ROC AUC score: 1.0

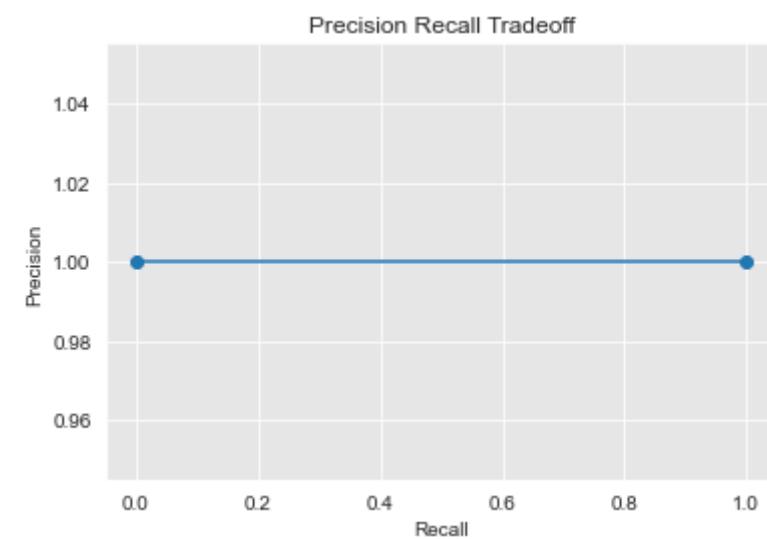
Class:N
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

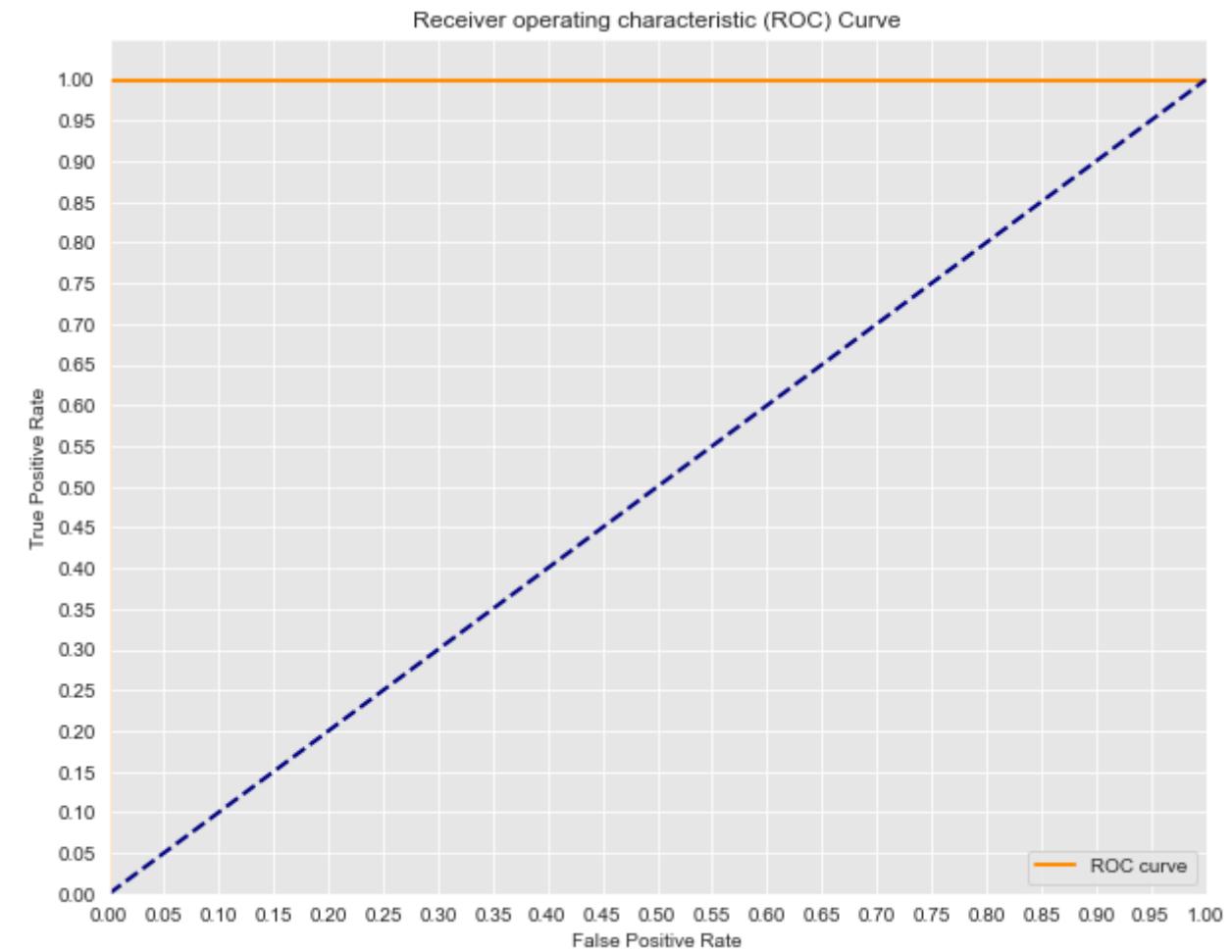




Cross Validated ROC AUC score: 1.0

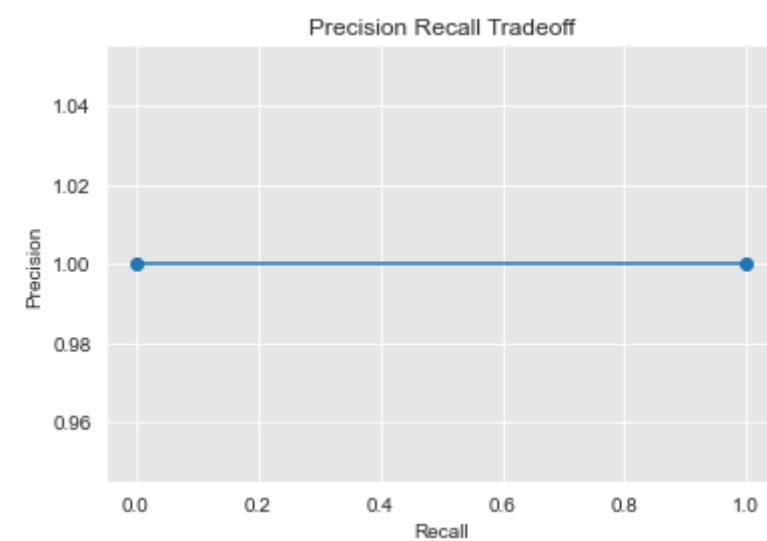
Class:P
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

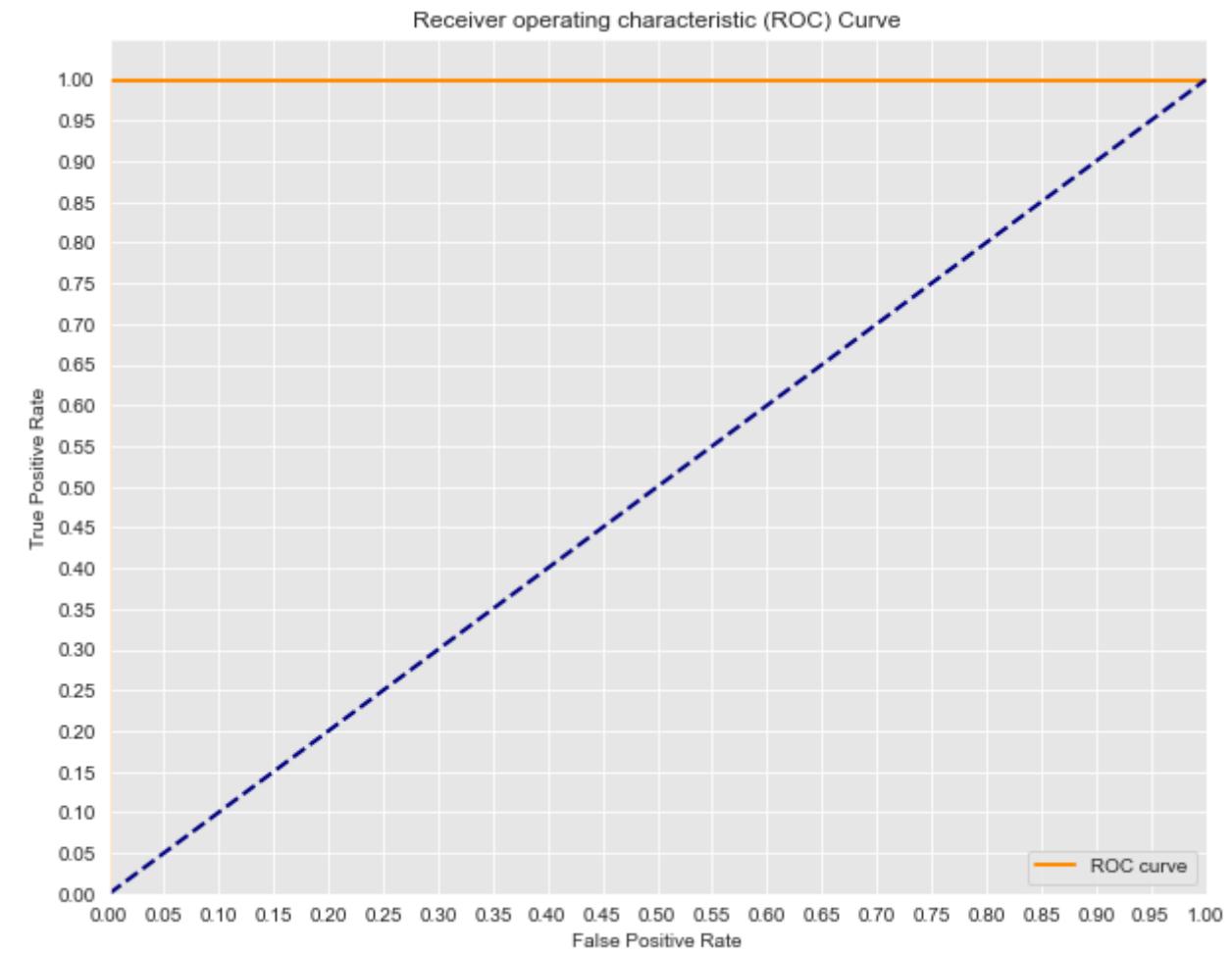




Cross Validated ROC AUC score: 1.0

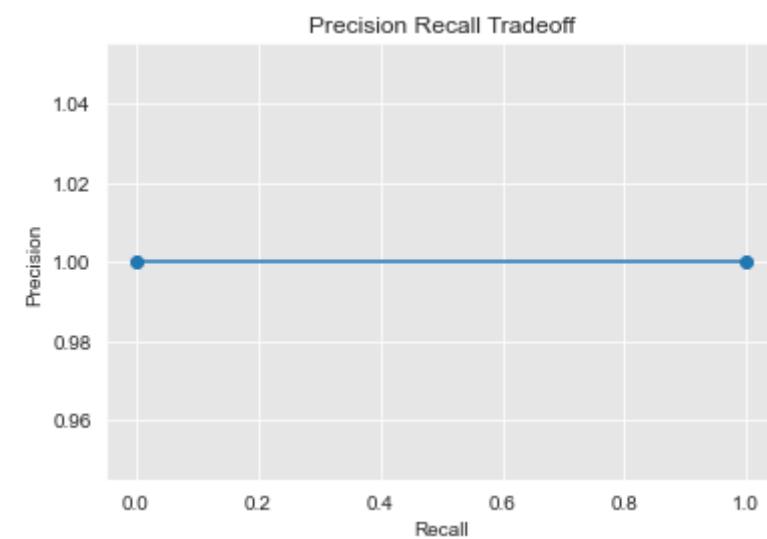
Class:Q
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

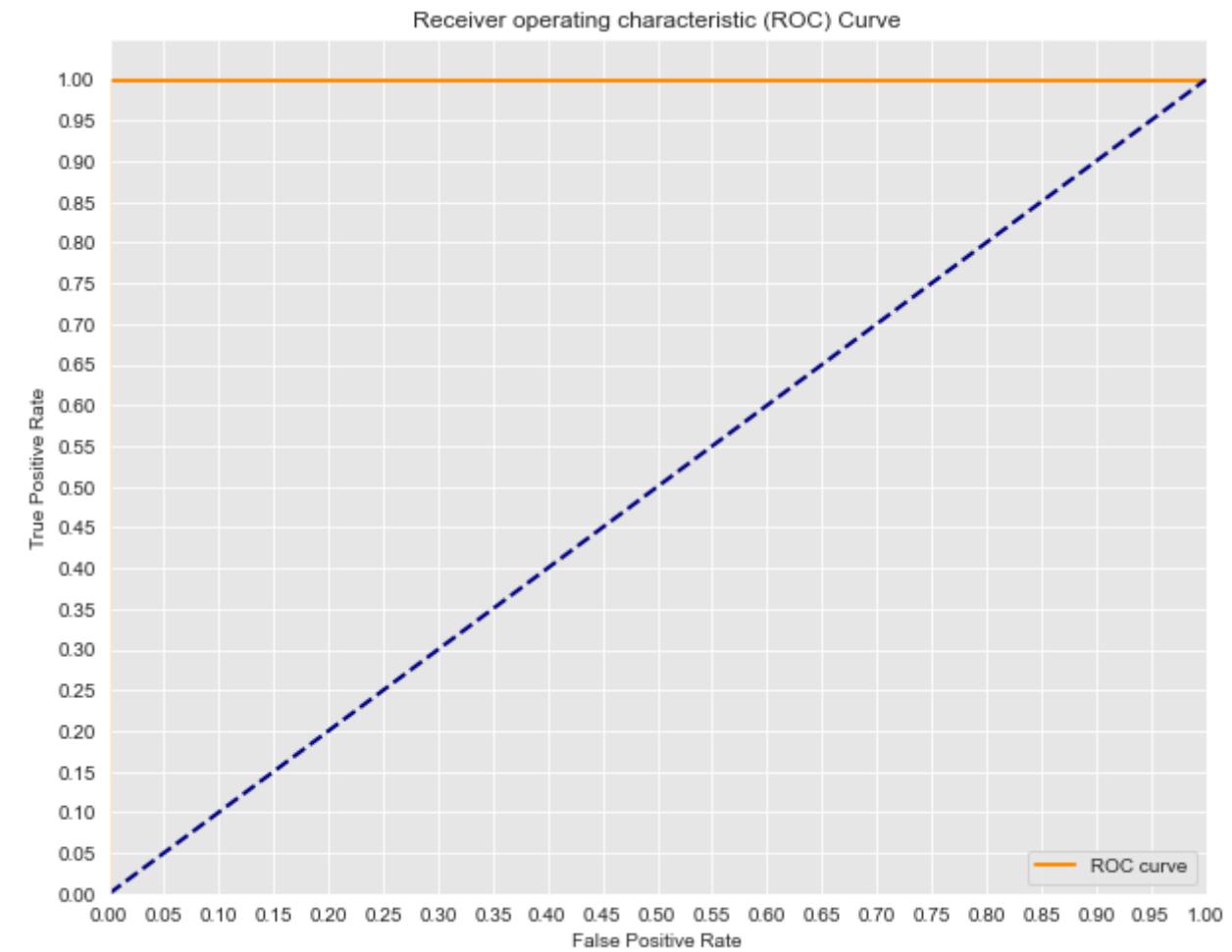




Cross Validated ROC AUC score: 1.0

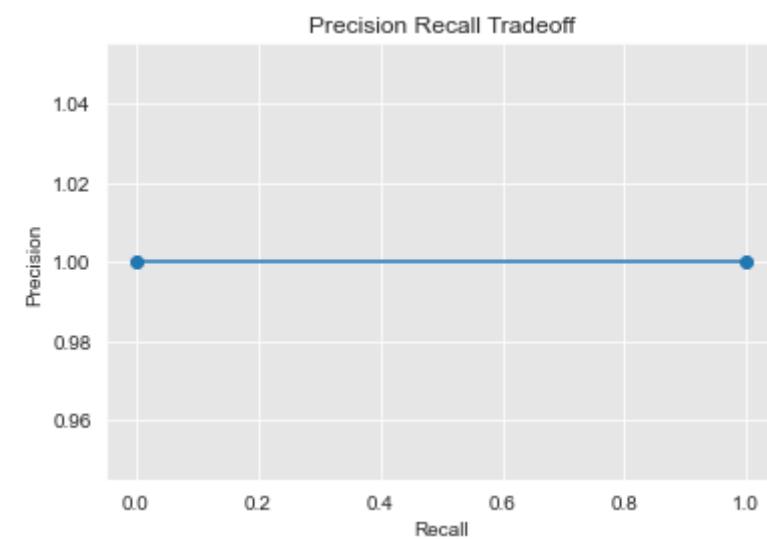
Class:R
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

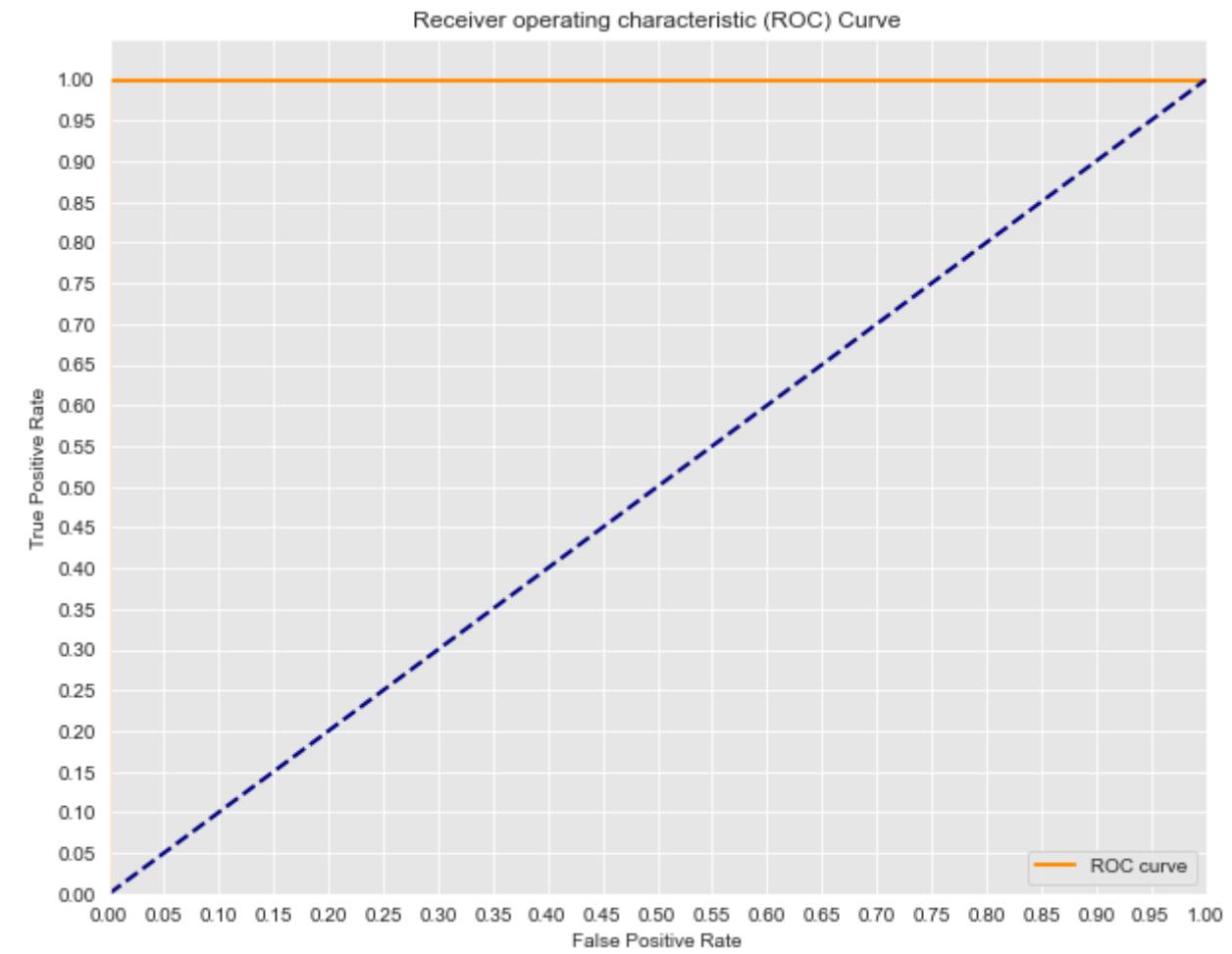




Cross Validated ROC AUC score: 1.0

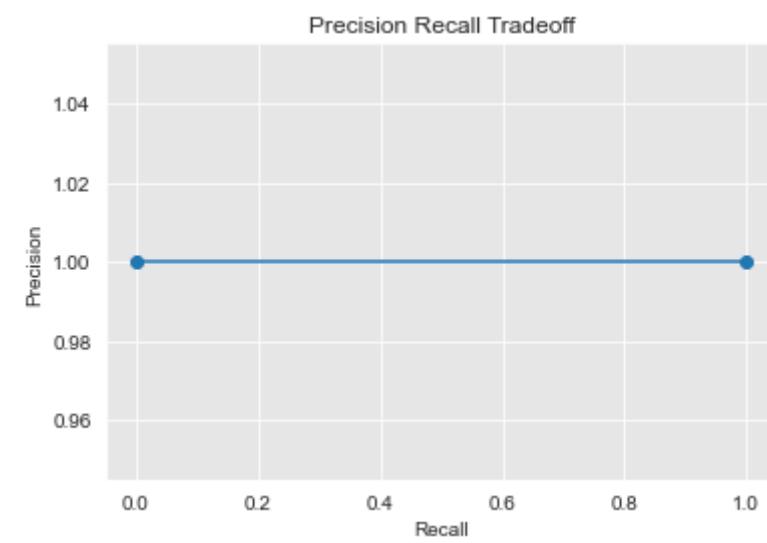
Class:S
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

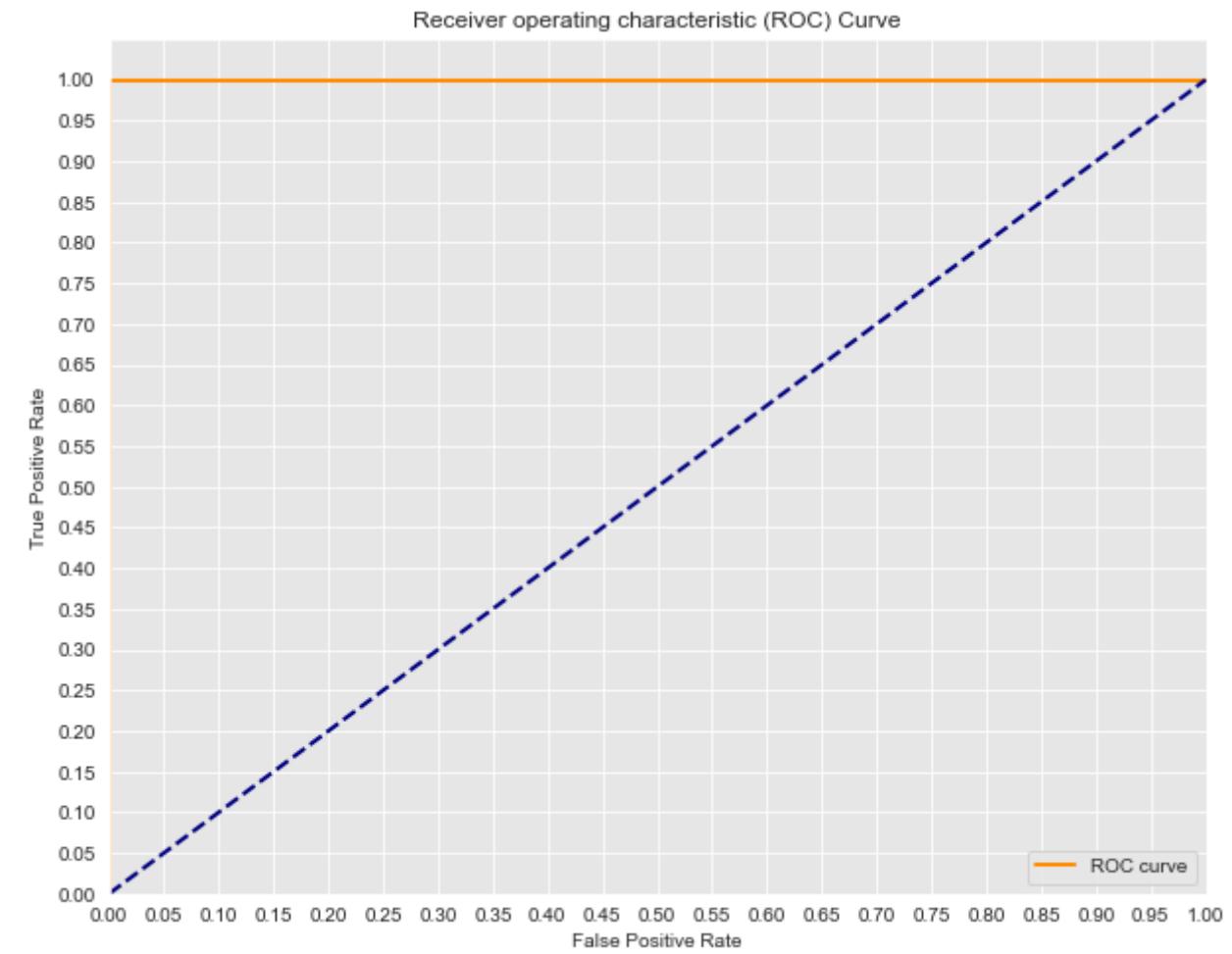




Cross Validated ROC AUC score: 1.0

Class:Stop
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

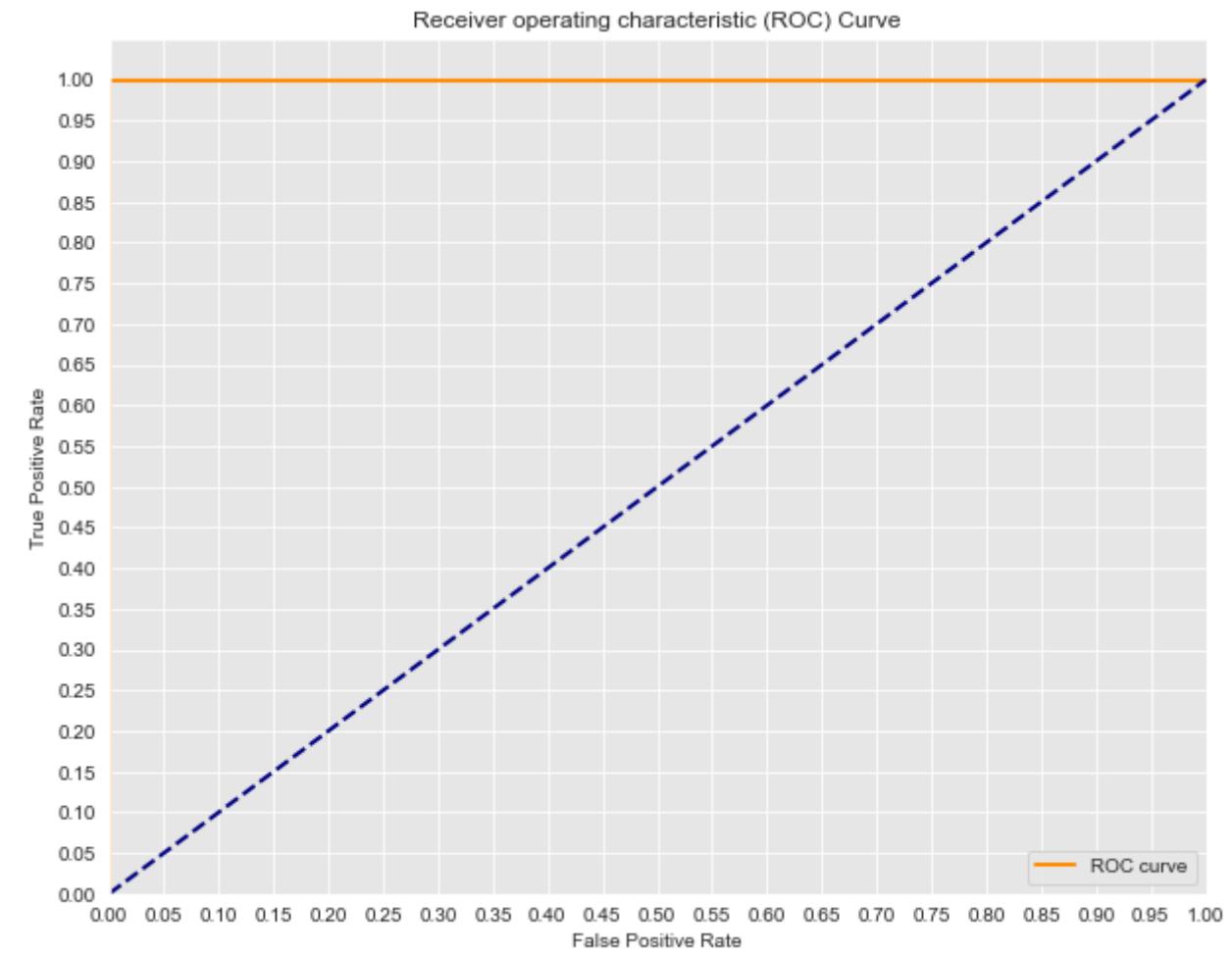




Cross Validated ROC AUC score: 1.0

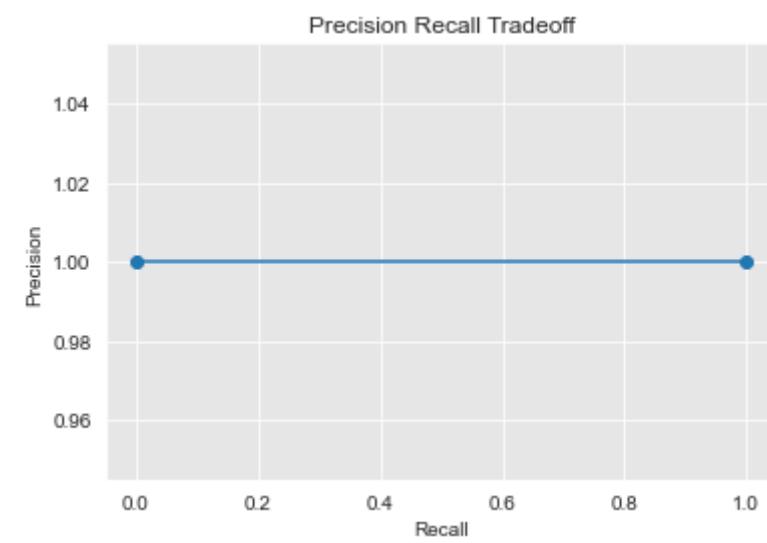
Class:T
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

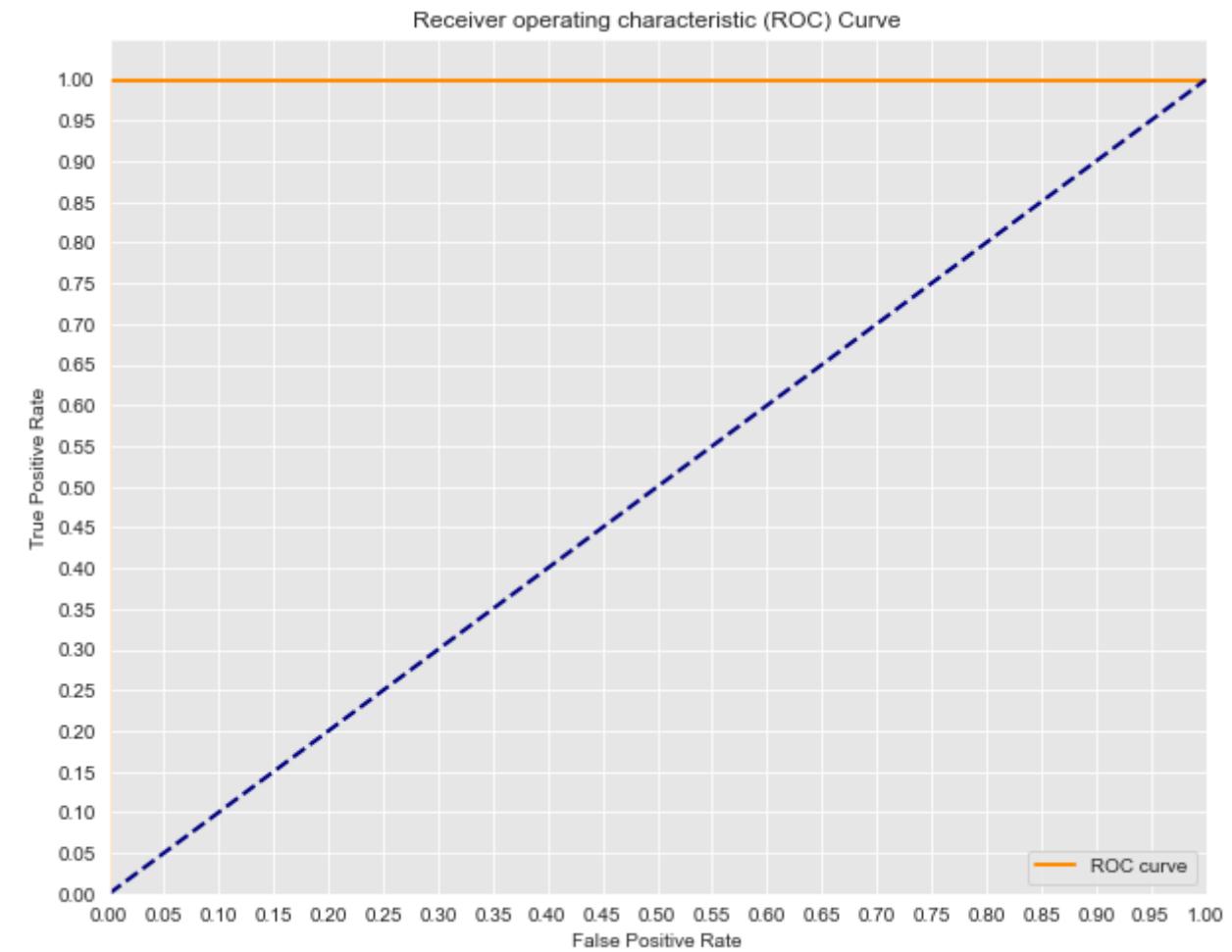




Cross Validated ROC AUC score: 1.0

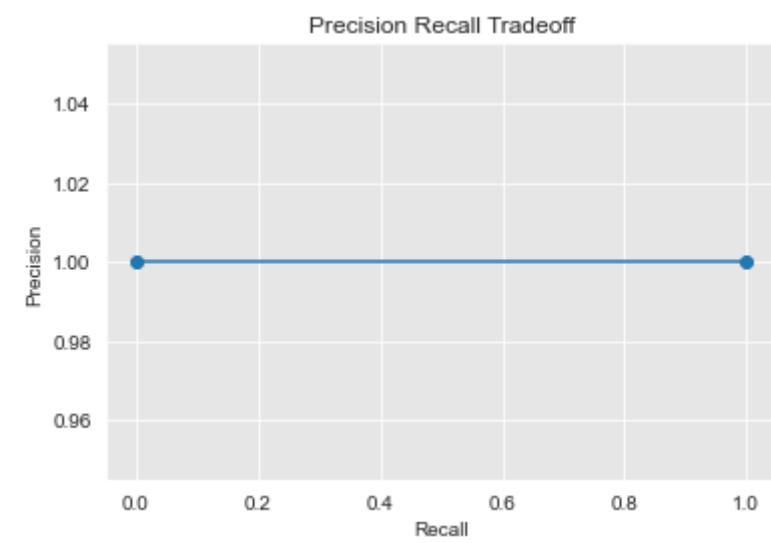
Class:V
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

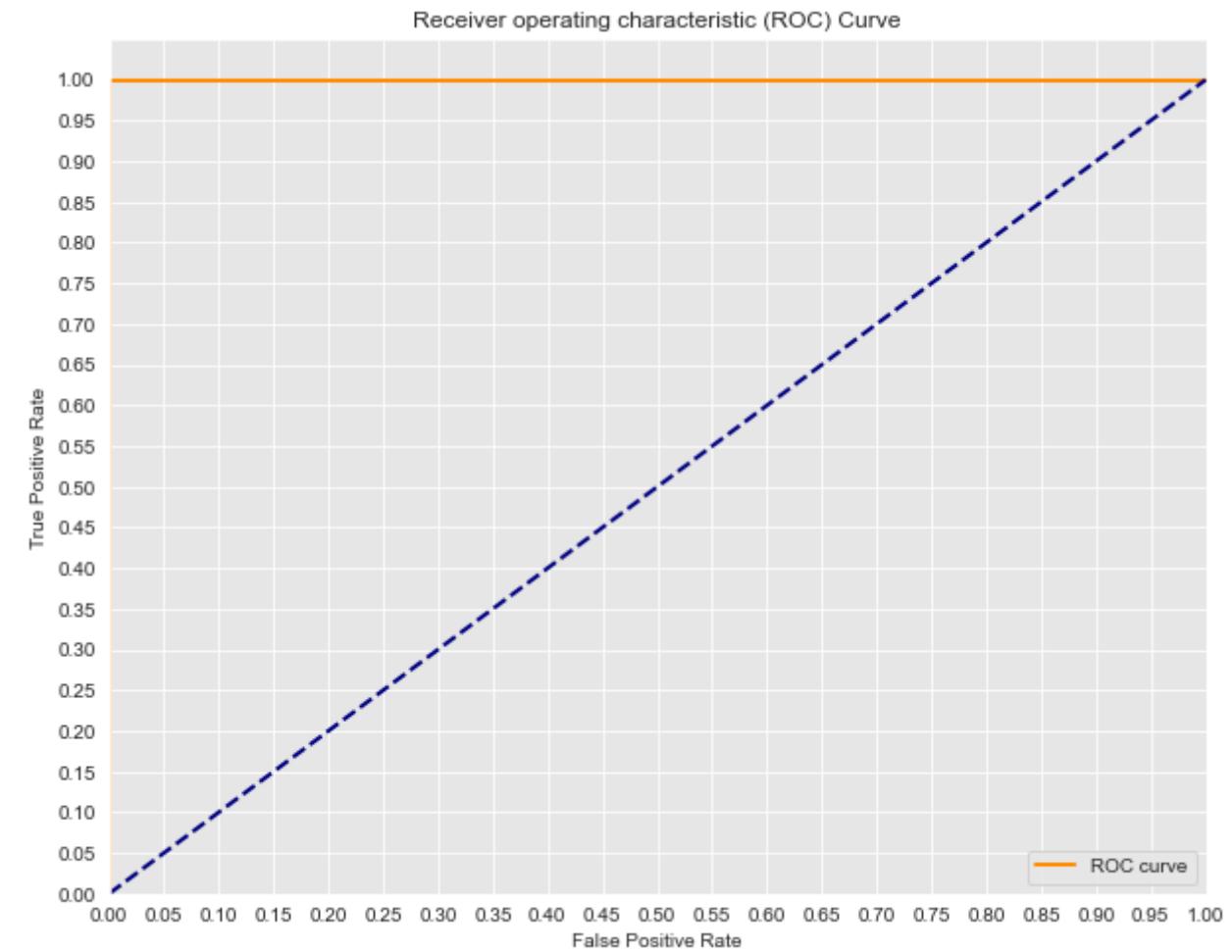




Cross Validated ROC AUC score: 1.0

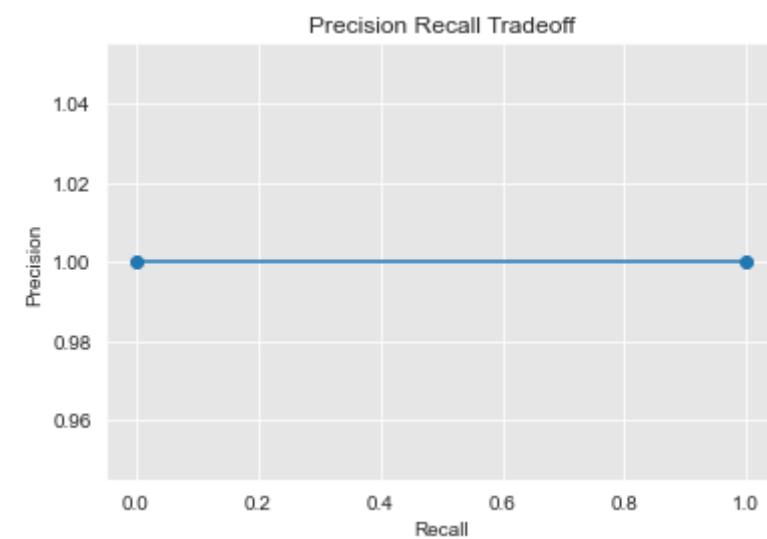
Class:W
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

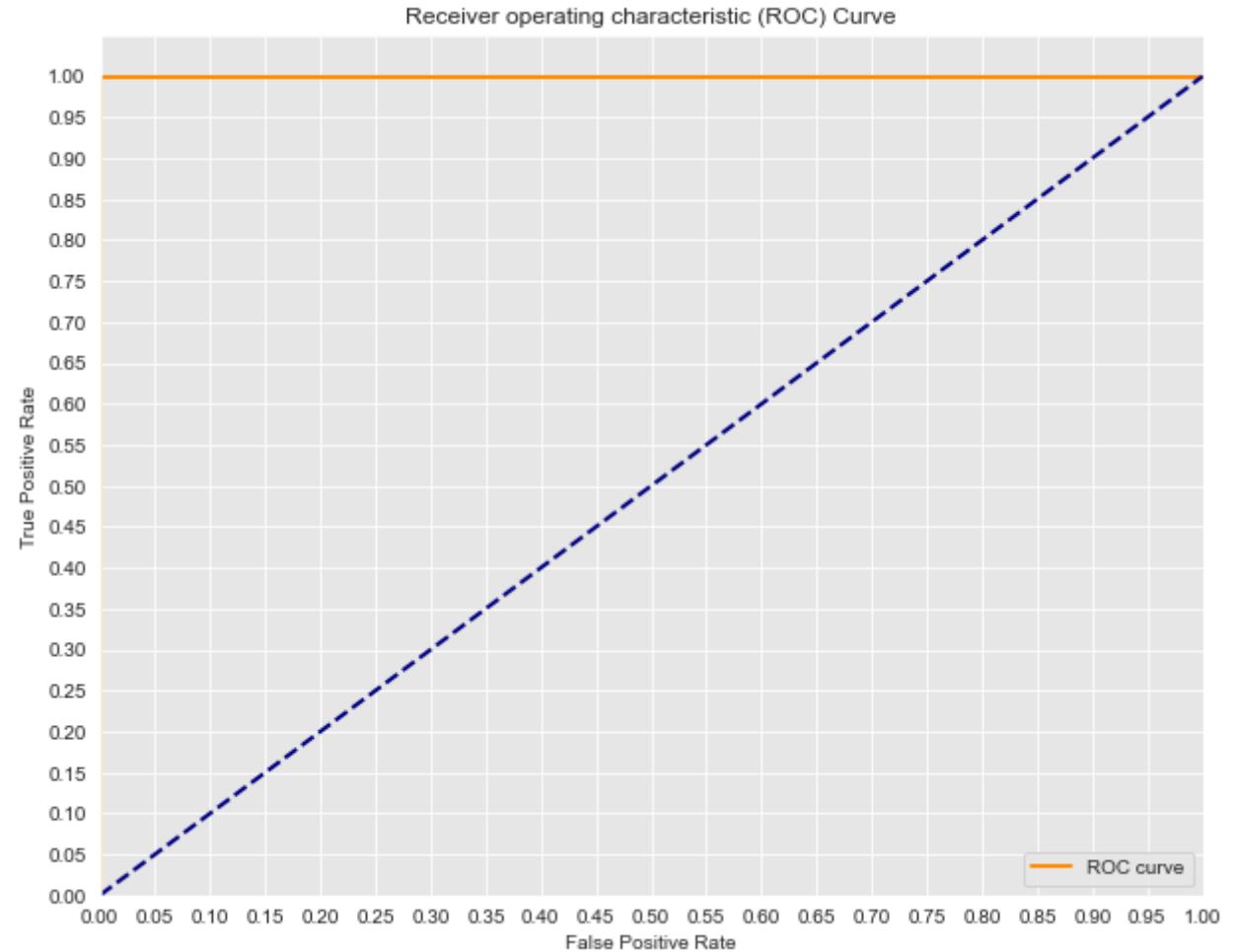




Cross Validated ROC AUC score: 1.0

Class:Y
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

Fit the model to the training data.

```
In [125]: random_forest.fit(x2_train, y2_train)
```

```
Out[125]: RandomForestClassifier()
```

Make predictions.

```
In [126]: y2_hat_train_rf = random_forest.predict(x2_train)
```

All predictions were classified correctly.

```
In [127]: con_mat(y2_train,y2_hat_train_rf)
```

Confusion Matrix

	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	\
0	370	0	0	0	0	0	0	0	0	0	...	0	0	0	0	\
1	0	182	0	0	0	0	0	0	0	0	...	0	0	0	0	\
2	0	0	273	0	0	0	0	0	0	0	...	0	0	0	0	\
3	0	0	0	447	0	0	0	0	0	0	...	0	0	0	0	\
4	0	0	0	0	833	0	0	0	0	0	...	0	0	0	0	\
5	0	0	0	0	0	720	0	0	0	0	...	0	0	0	0	\
6	0	0	0	0	0	0	142	0	0	0	...	0	0	0	0	\
7	0	0	0	0	0	0	0	315	0	0	...	0	0	0	0	\
8	0	0	0	0	0	0	0	0	485	0	...	0	0	0	0	\
9	0	0	0	0	0	0	0	0	0	437	...	0	0	0	0	\
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	\

```

11   0   0   0   0   0   0   0   0   0   0   0   ... 259   0   0   0
12   0   0   0   0   0   0   0   0   0   0   0   ... 0   209   0   0
13   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   200   0
14   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   824
15   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
16   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
17   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
18   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
19   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
20   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0

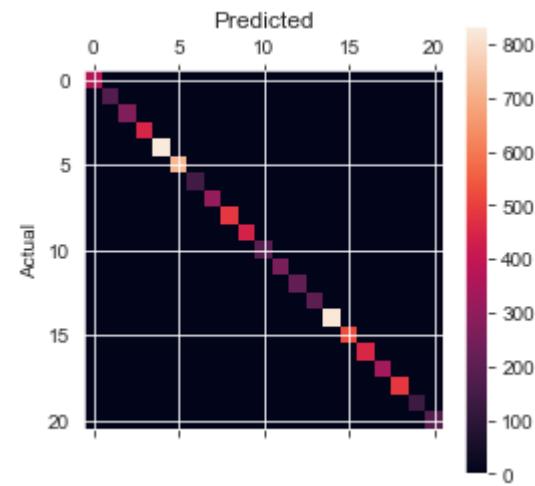
```

```

15  16  17  18  19  20
0   0   0   0   0   0
1   0   0   0   0   0
2   0   0   0   0   0
3   0   0   0   0   0
4   0   0   0   0   0
5   0   0   0   0   0
6   0   0   0   0   0
7   0   0   0   0   0
8   0   0   0   0   0
9   0   0   0   0   0
10  0   0   0   0   0
11  0   0   0   0   0
12  0   0   0   0   0
13  0   0   0   0   0
14  0   0   0   0   0
15  522  0   0   0   0
16  0   451  0   0   0
17  0   0   339  0   0
18  0   0   0   488  0
19  0   0   0   0   134
20  0   0   0   0   0   183

```

[21 rows x 21 columns]



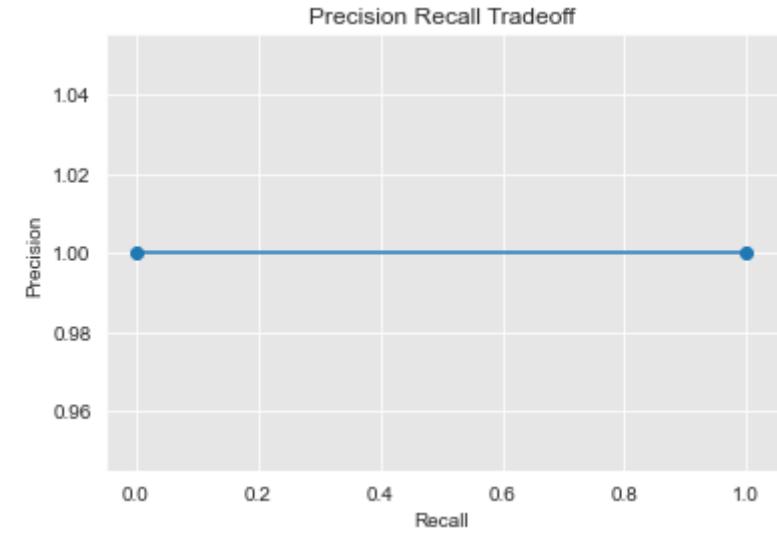
Separate predicted values into each class for metrics.

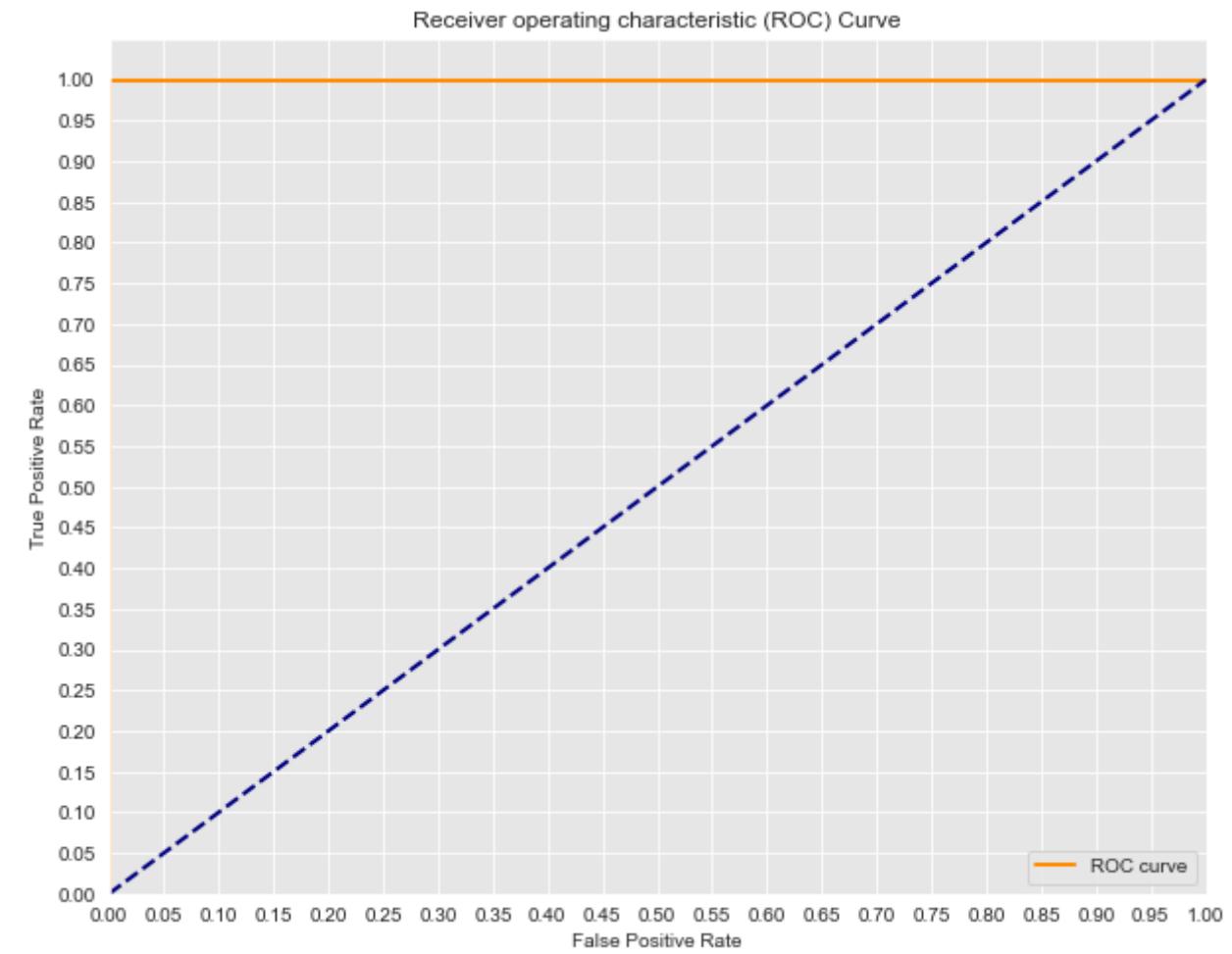
```
In [135...]
predictions_class0=y2_hat_train_rf==0
predictions_class1=y2_hat_train_rf==1
predictions_class2=y2_hat_train_rf==2
predictions_class3=y2_hat_train_rf==3
predictions_class4=y2_hat_train_rf==4
predictions_class5=y2_hat_train_rf==5
predictions_class6=y2_hat_train_rf==6
predictions_class7=y2_hat_train_rf==7
predictions_class8=y2_hat_train_rf==8
```

```
predictions_class9=y2_hat_train_rf==9
predictions_class10=y2_hat_train_rf==10
predictions_class11=y2_hat_train_rf==11
predictions_class12=y2_hat_train_rf==12
predictions_class13=y2_hat_train_rf==13
predictions_class14=y2_hat_train_rf==14
predictions_class15=y2_hat_train_rf==15
predictions_class16=y2_hat_train_rf==16
predictions_class17=y2_hat_train_rf==17
predictions_class18=y2_hat_train_rf==18
predictions_class19=y2_hat_train_rf==19
predictions_class20=y2_hat_train_rf==20
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3,predictions_class4,predictions_class5,predictions_class6
    ,predictions_class7,predictions_class8,predictions_class9,predictions_class10,predictions_class11,predictions_class12
    ,predictions_class13,predictions_class14,predictions_class15,predictions_class16,predictions_class17,predictions_class18
    ,predictions_class19,predictions_class20]
```

```
In [136]: multiclass(random_forest,x2_train,a,p,translation_df2['Amino_Acids_copy'])
```

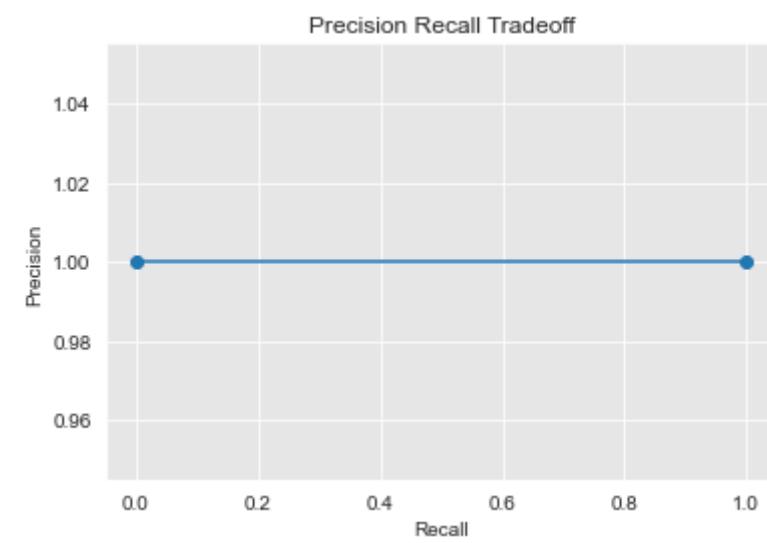
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

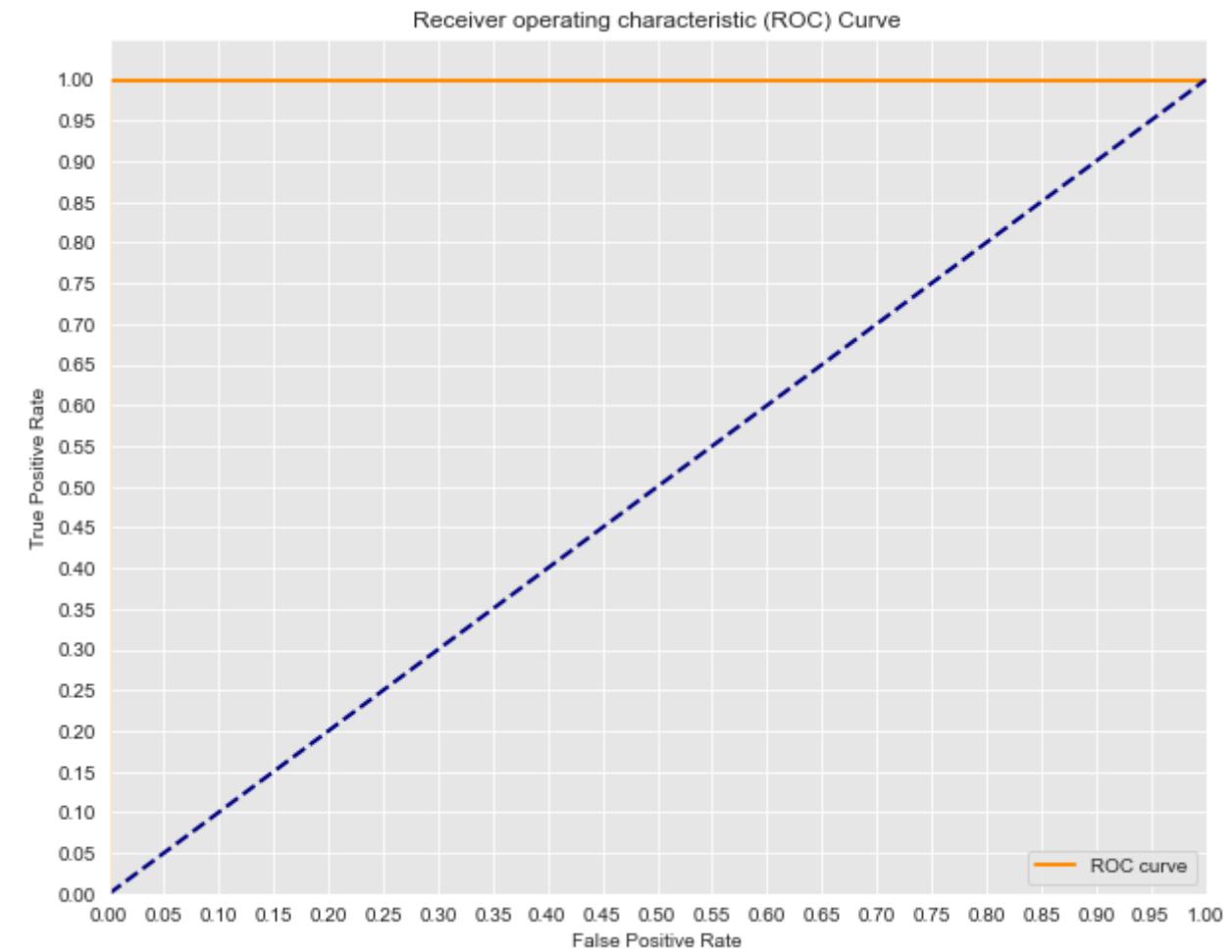




Cross Validated ROC AUC score: 1.0

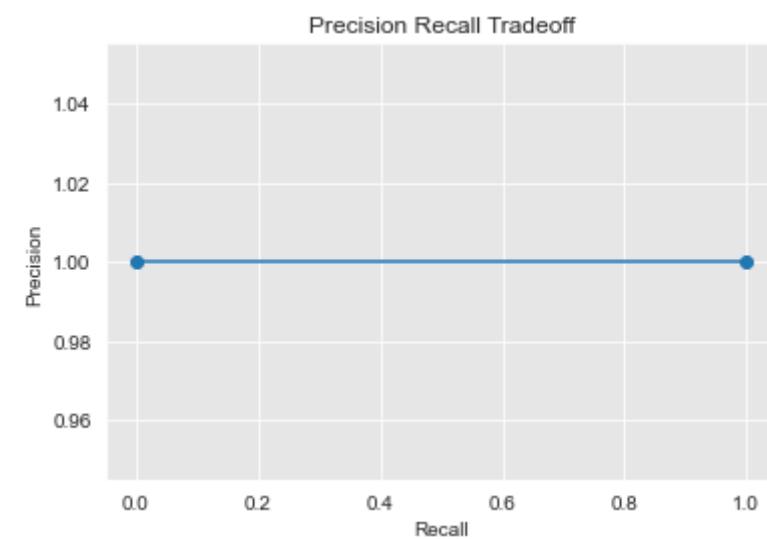
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

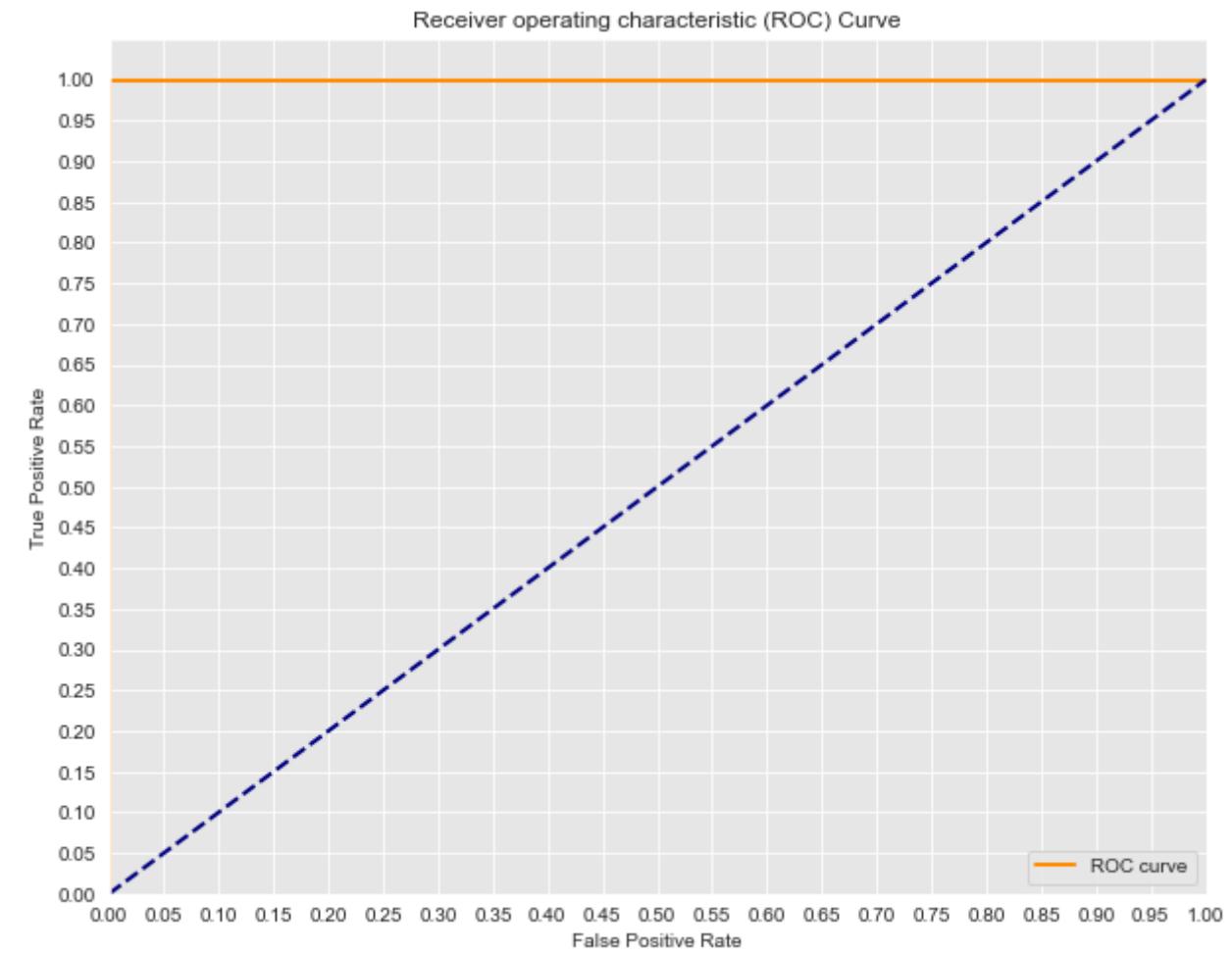




Cross Validated ROC AUC score: 1.0

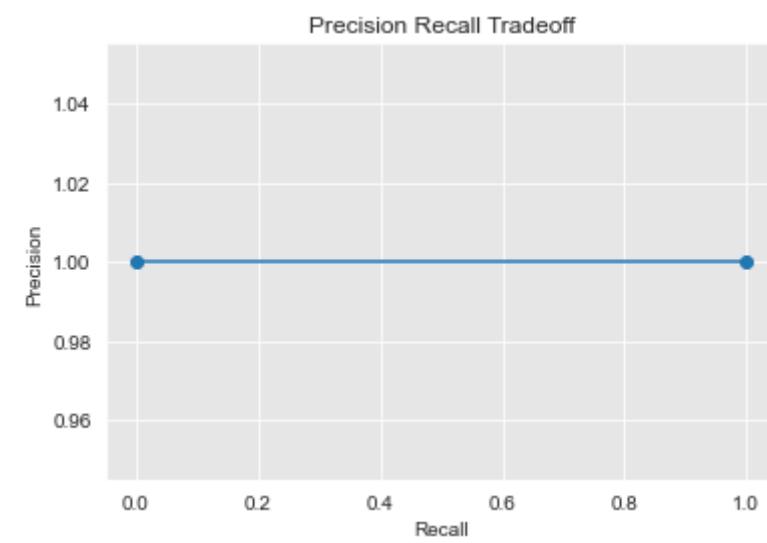
Class:D
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

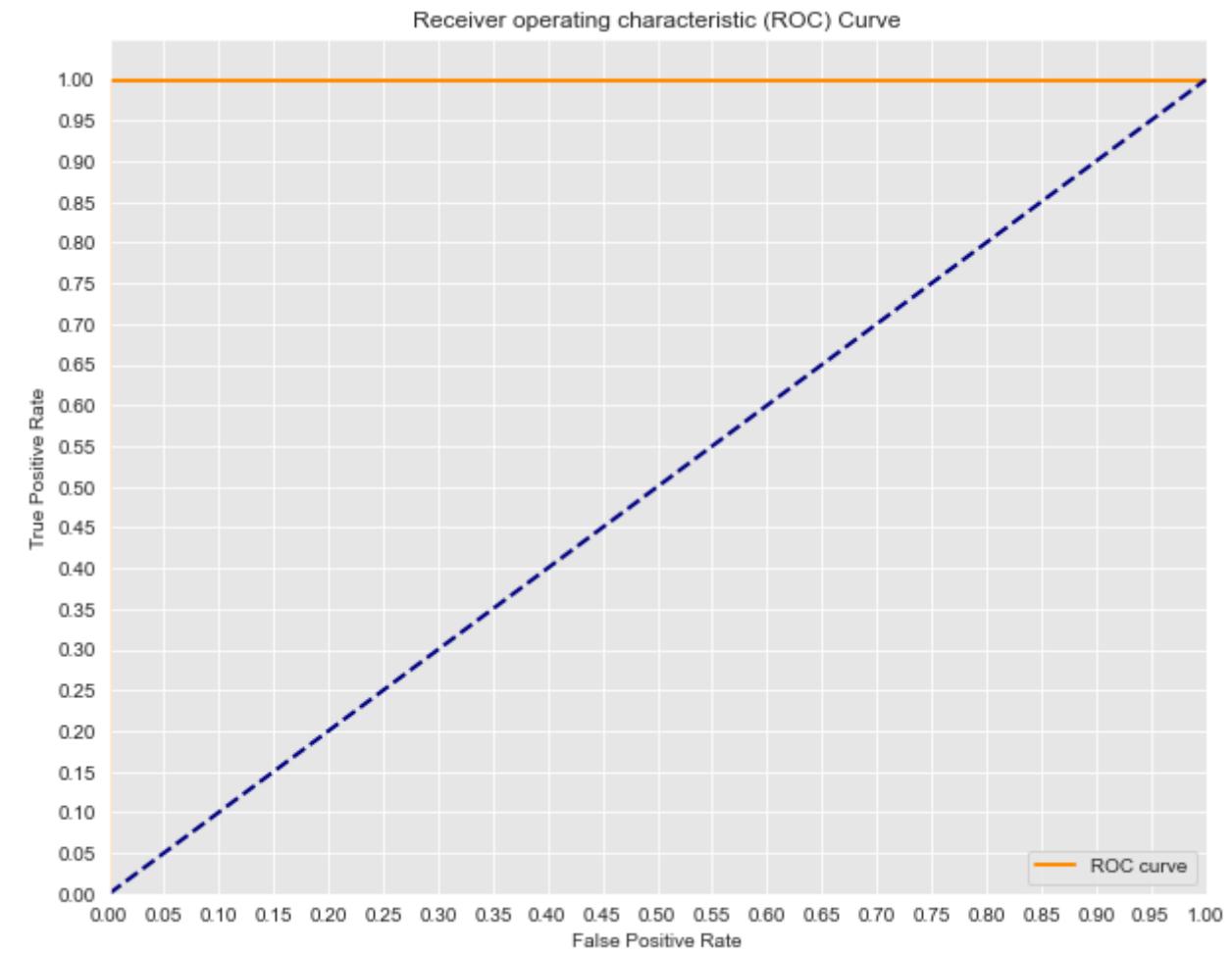




Cross Validated ROC AUC score: 1.0

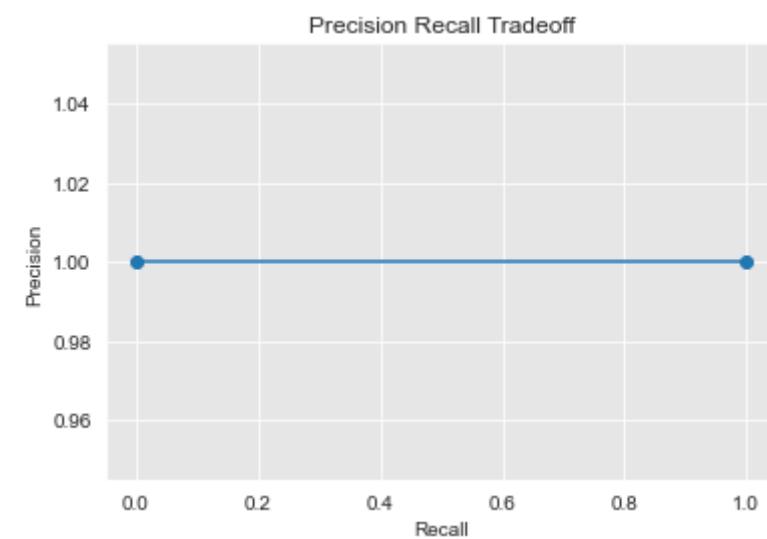
Class:E
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

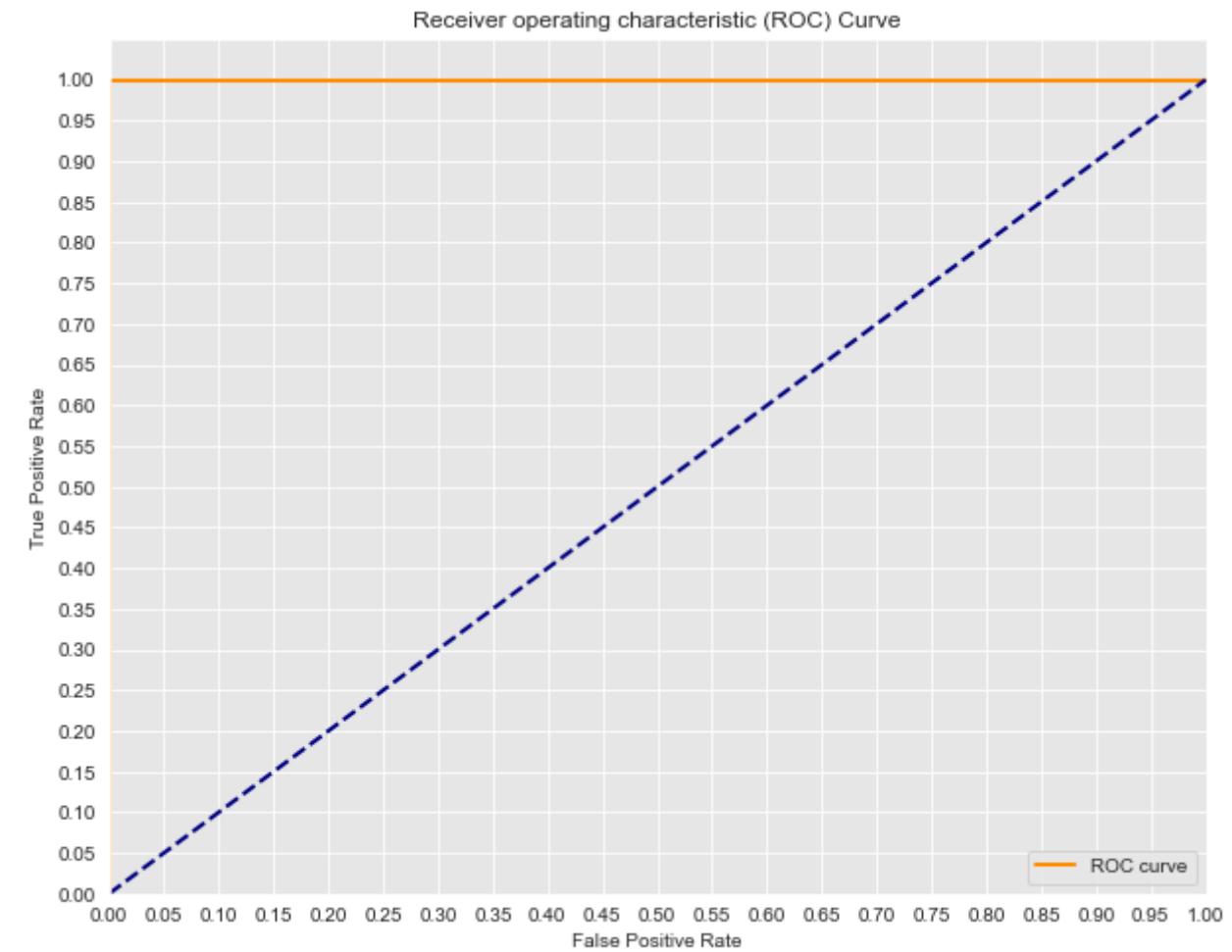




Cross Validated ROC AUC score: 1.0

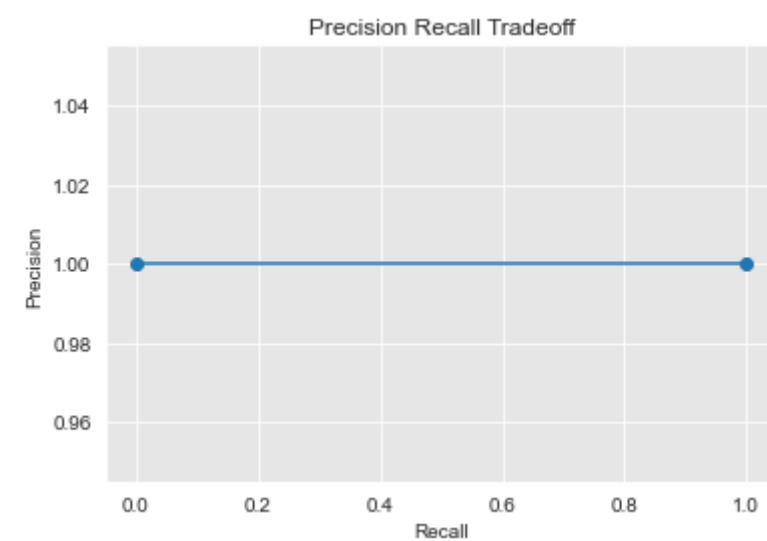
Class:F
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

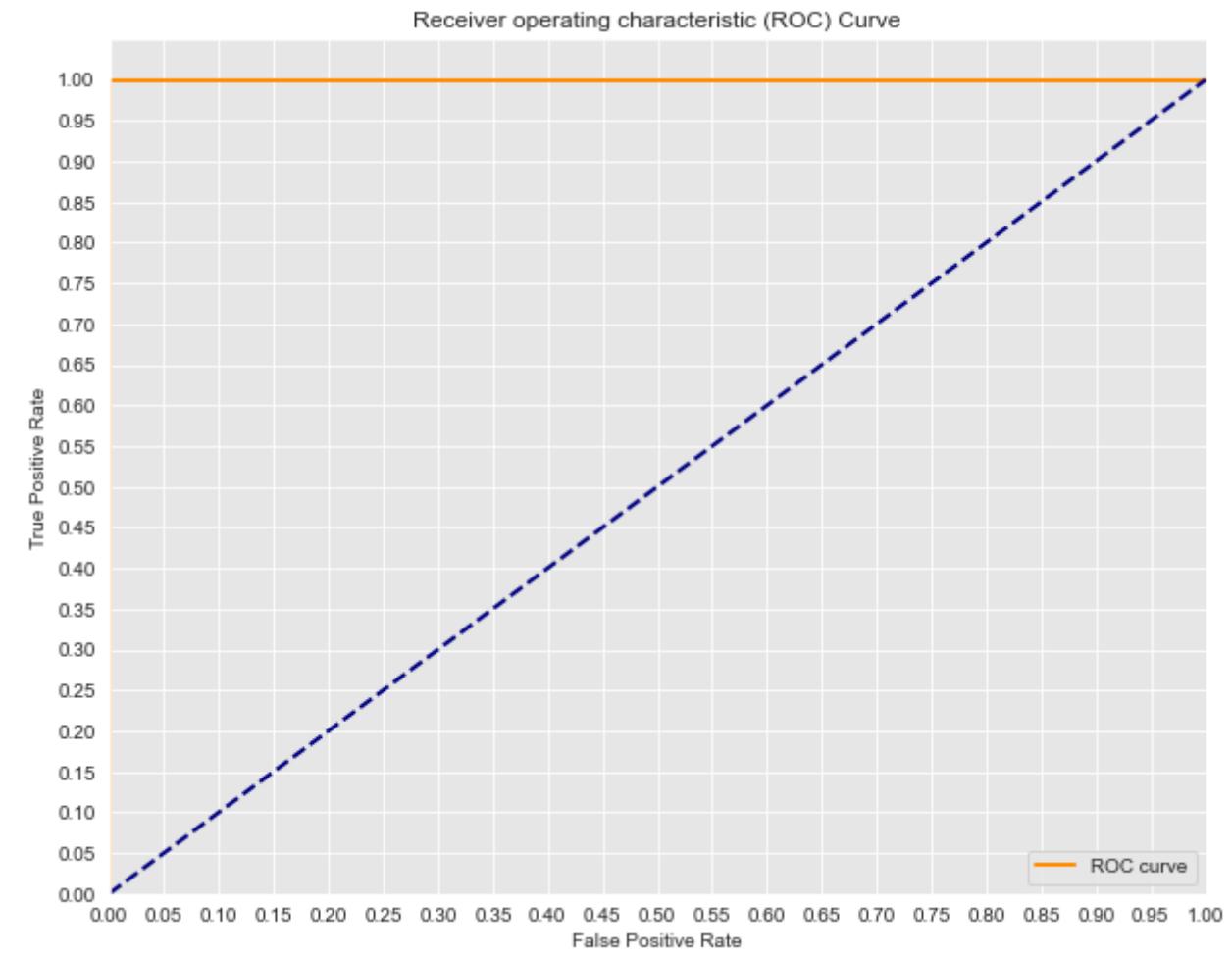




Cross Validated ROC AUC score: 1.0

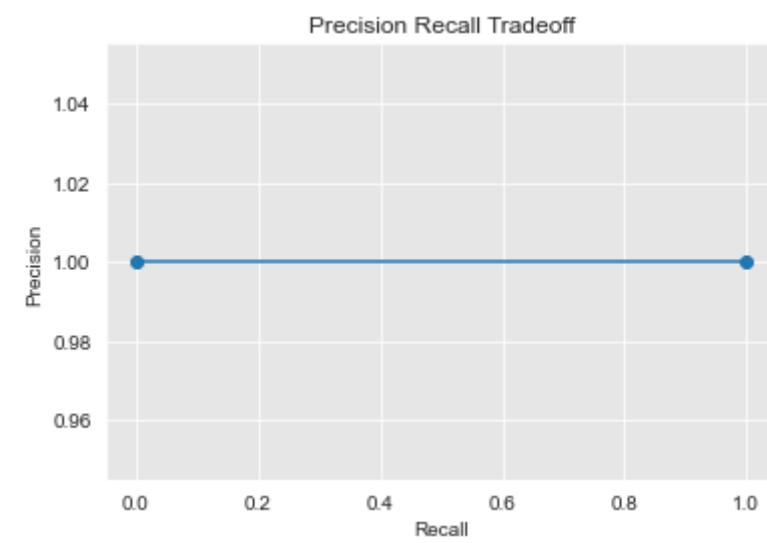
Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

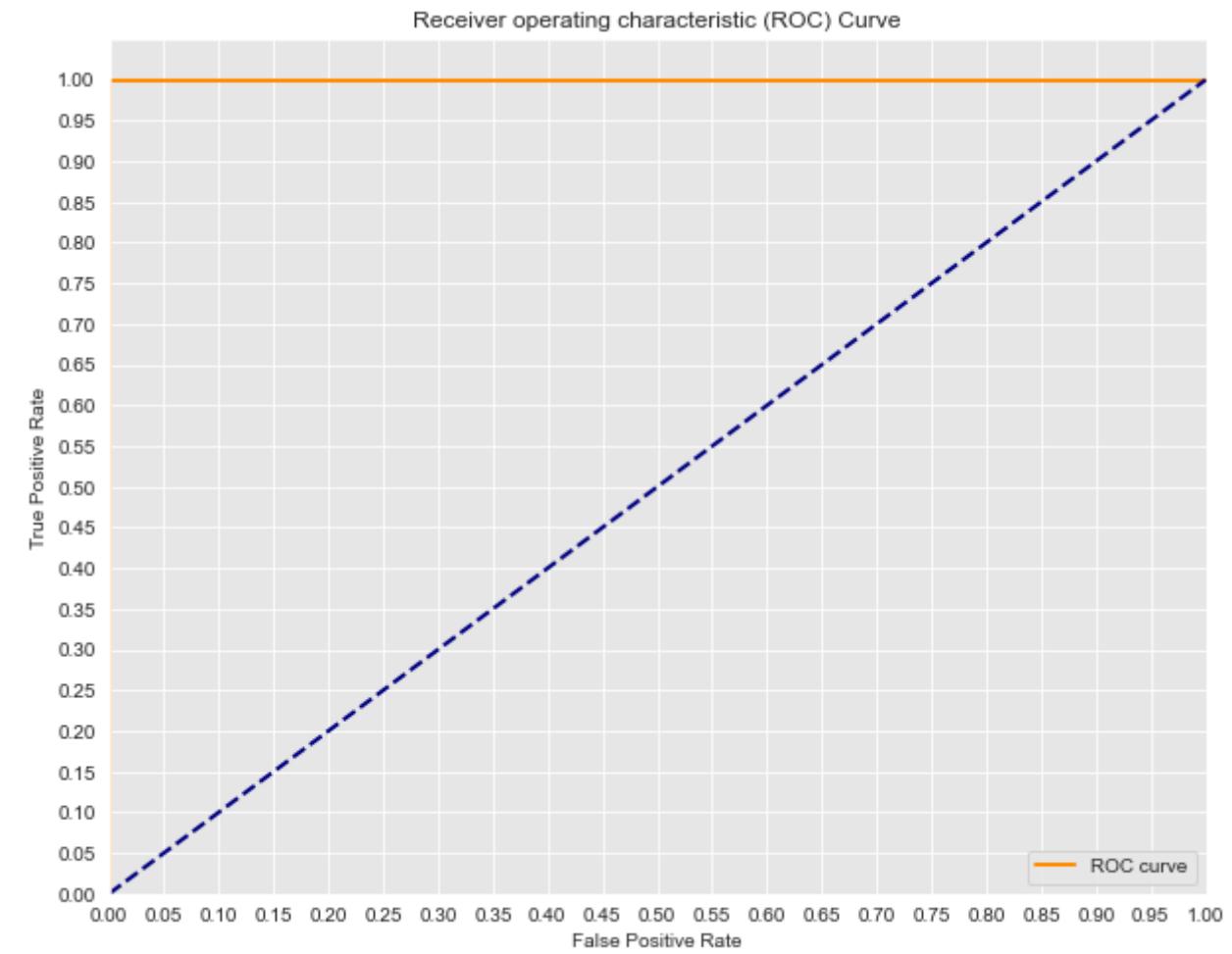




Cross Validated ROC AUC score: 1.0

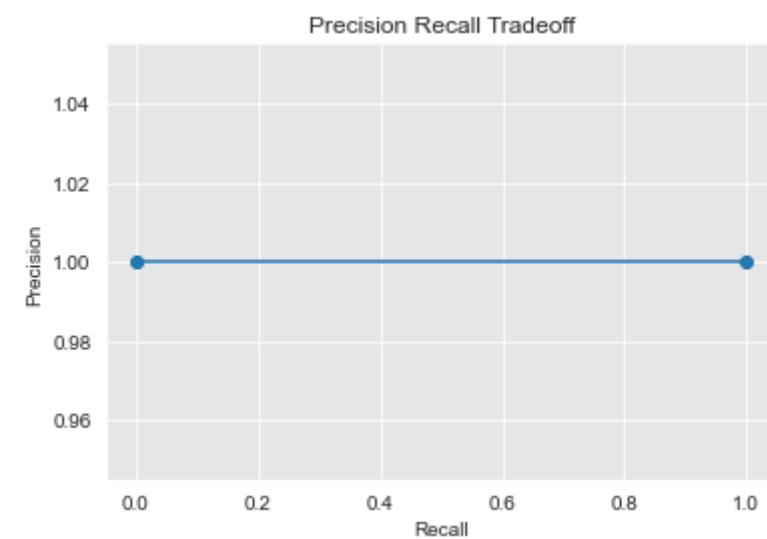
Class:H
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

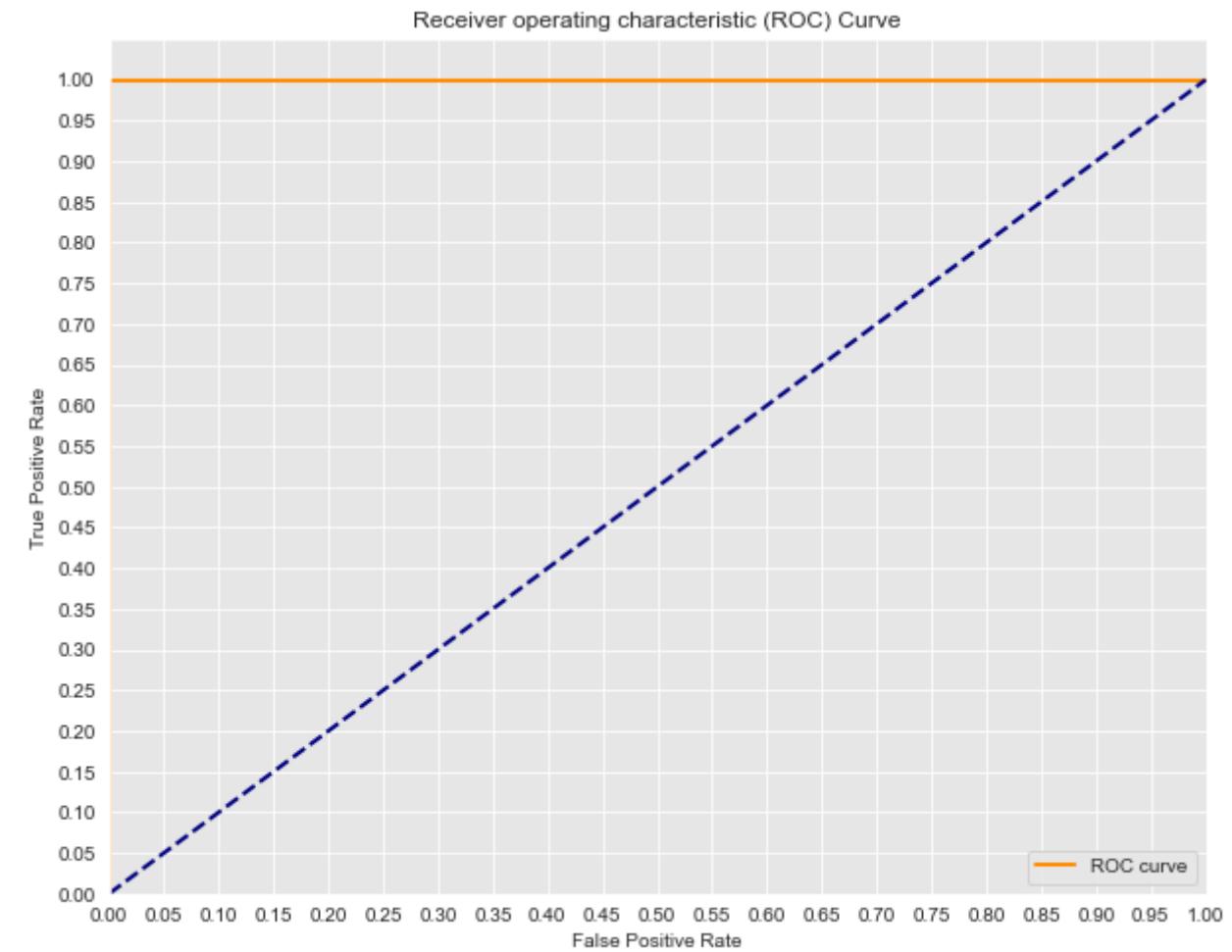




Cross Validated ROC AUC score: 1.0

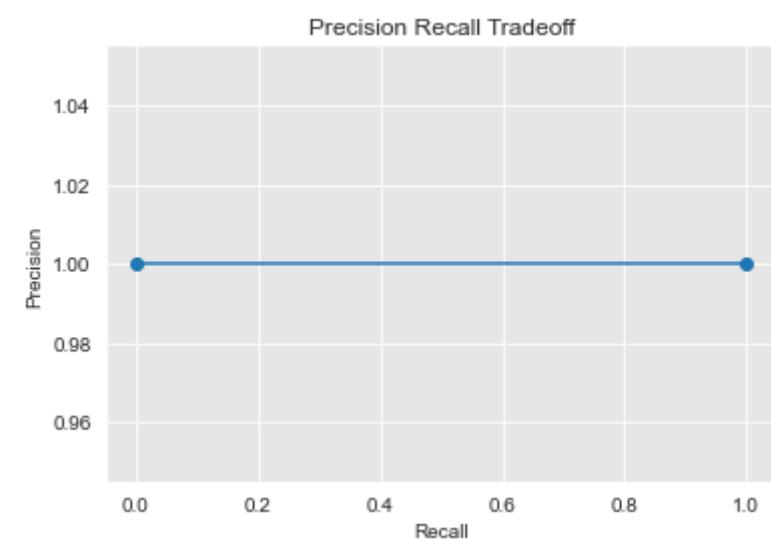
Class:I
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

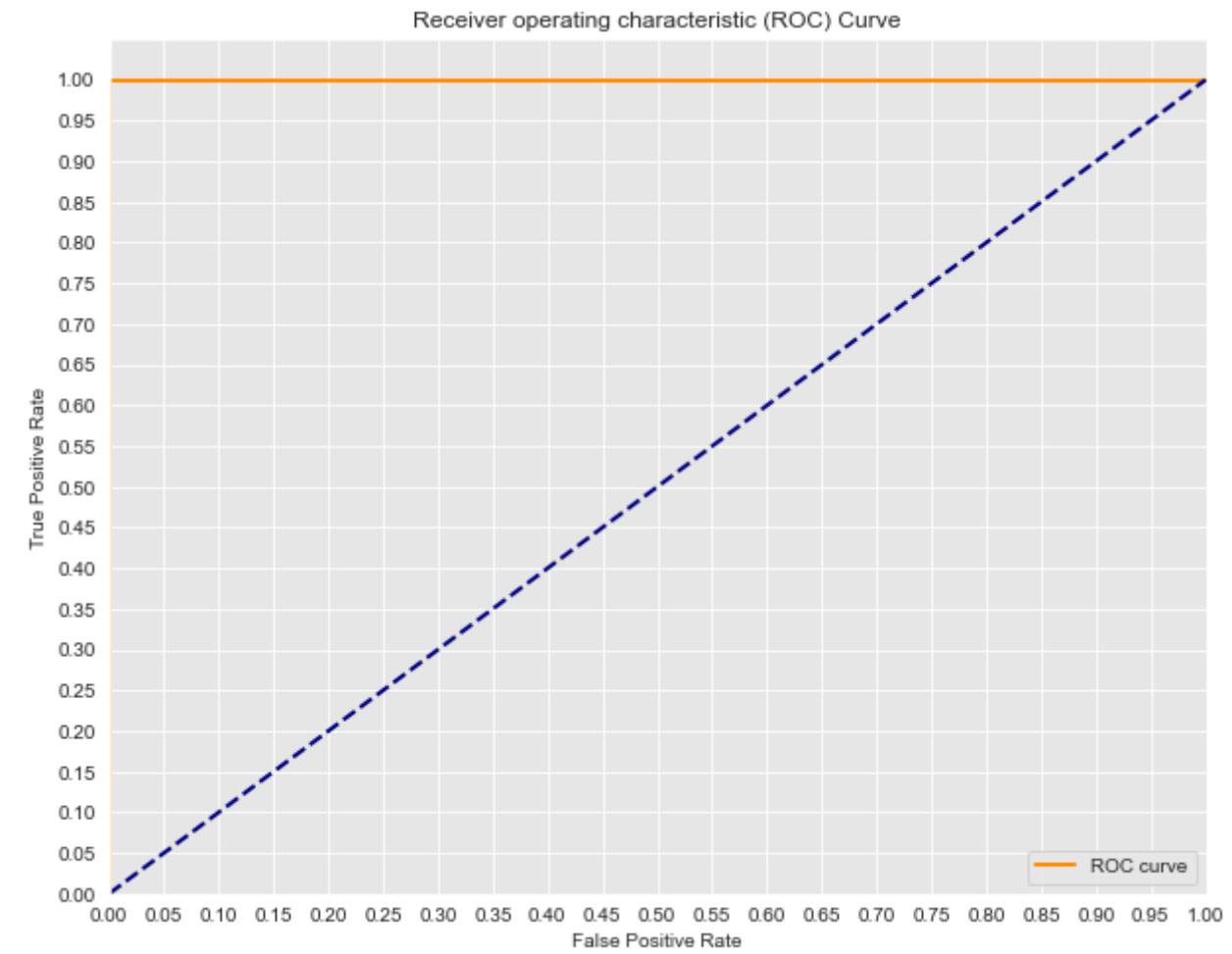




Cross Validated ROC AUC score: 1.0

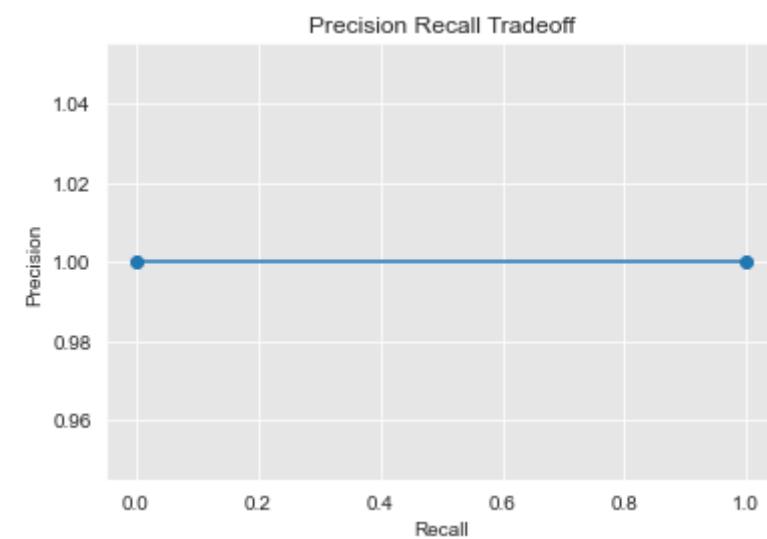
Class:K
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

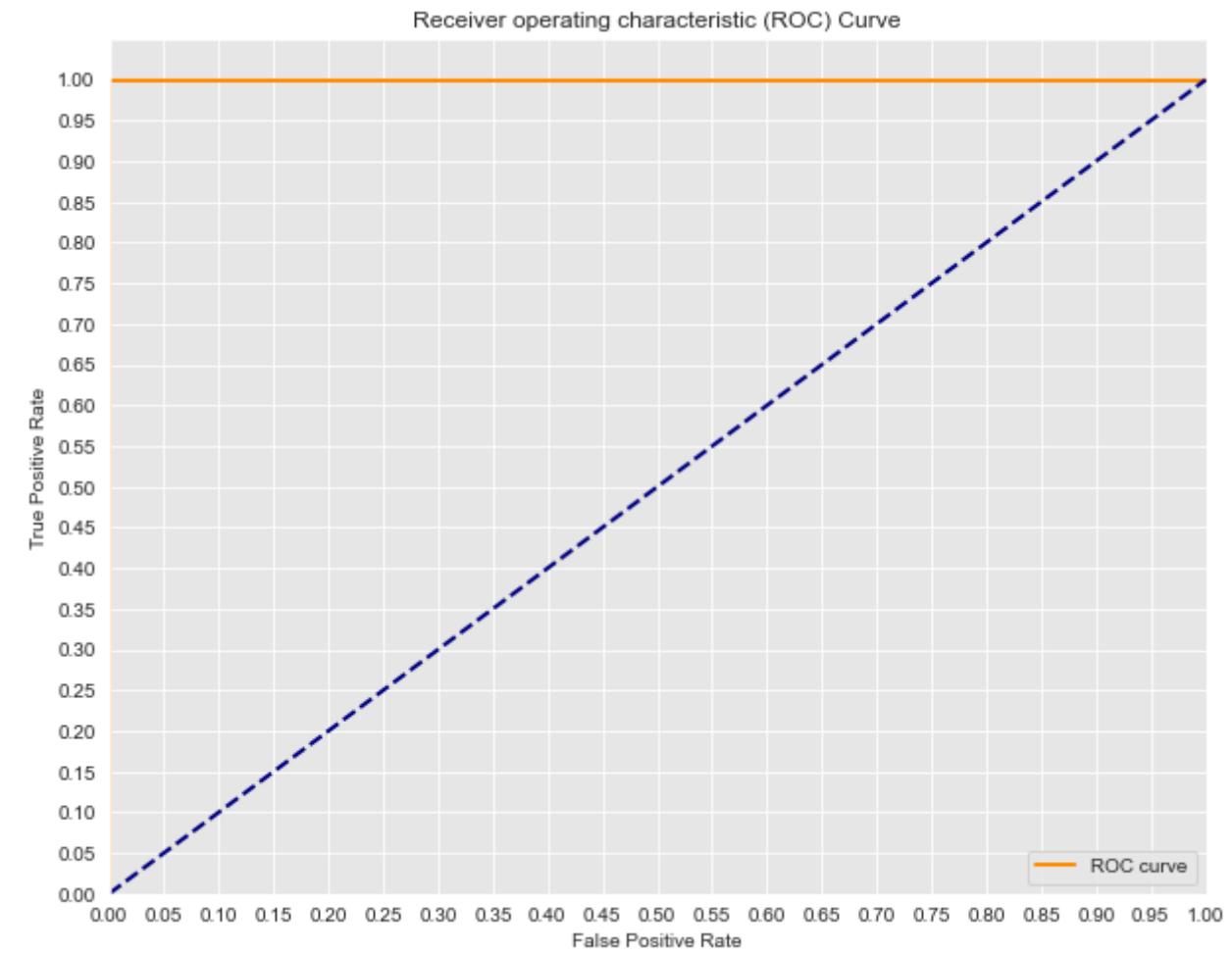




Cross Validated ROC AUC score: 1.0

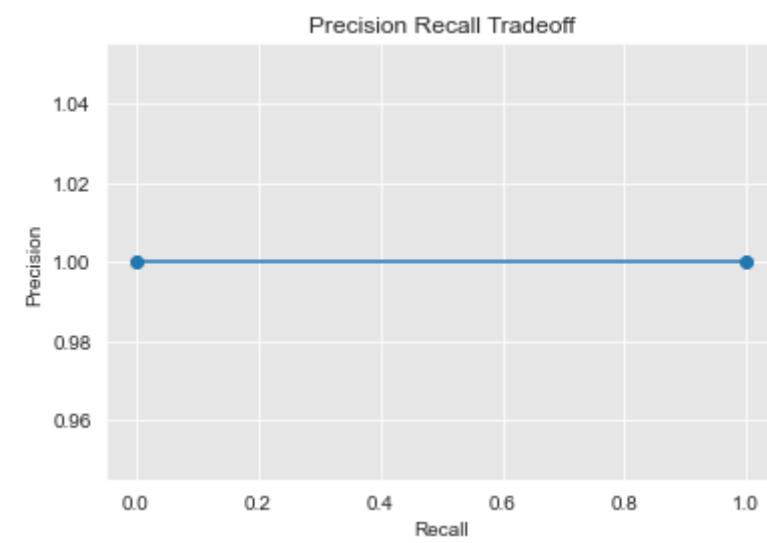
Class:L
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

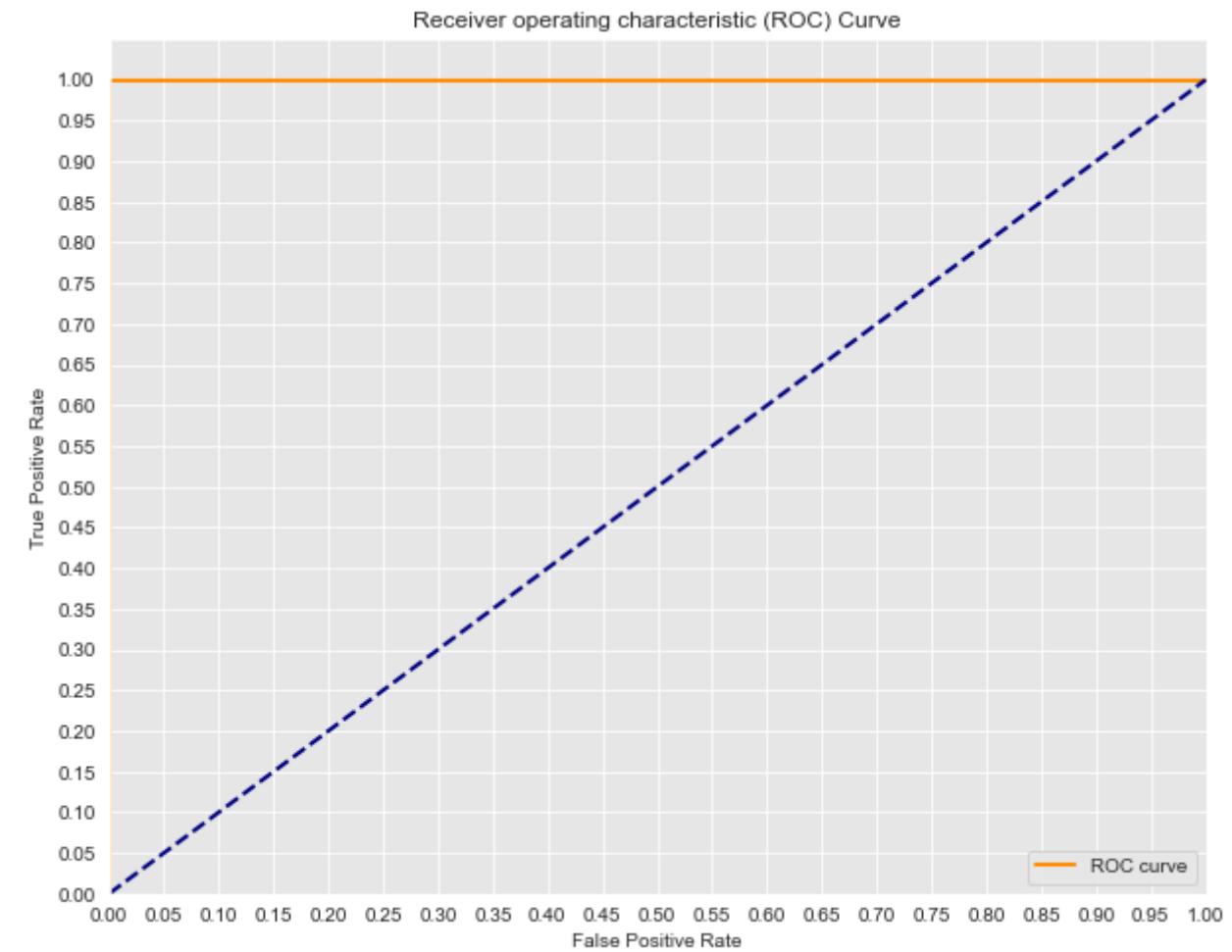




Cross Validated ROC AUC score: 1.0

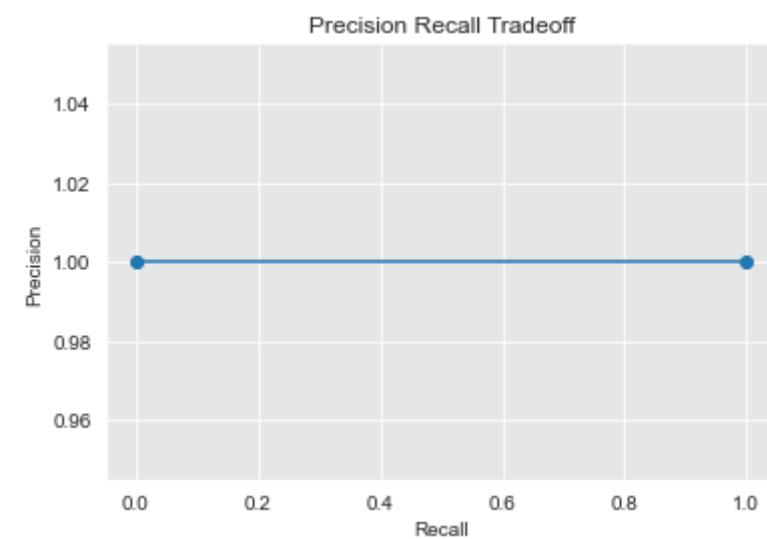
Class:M
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

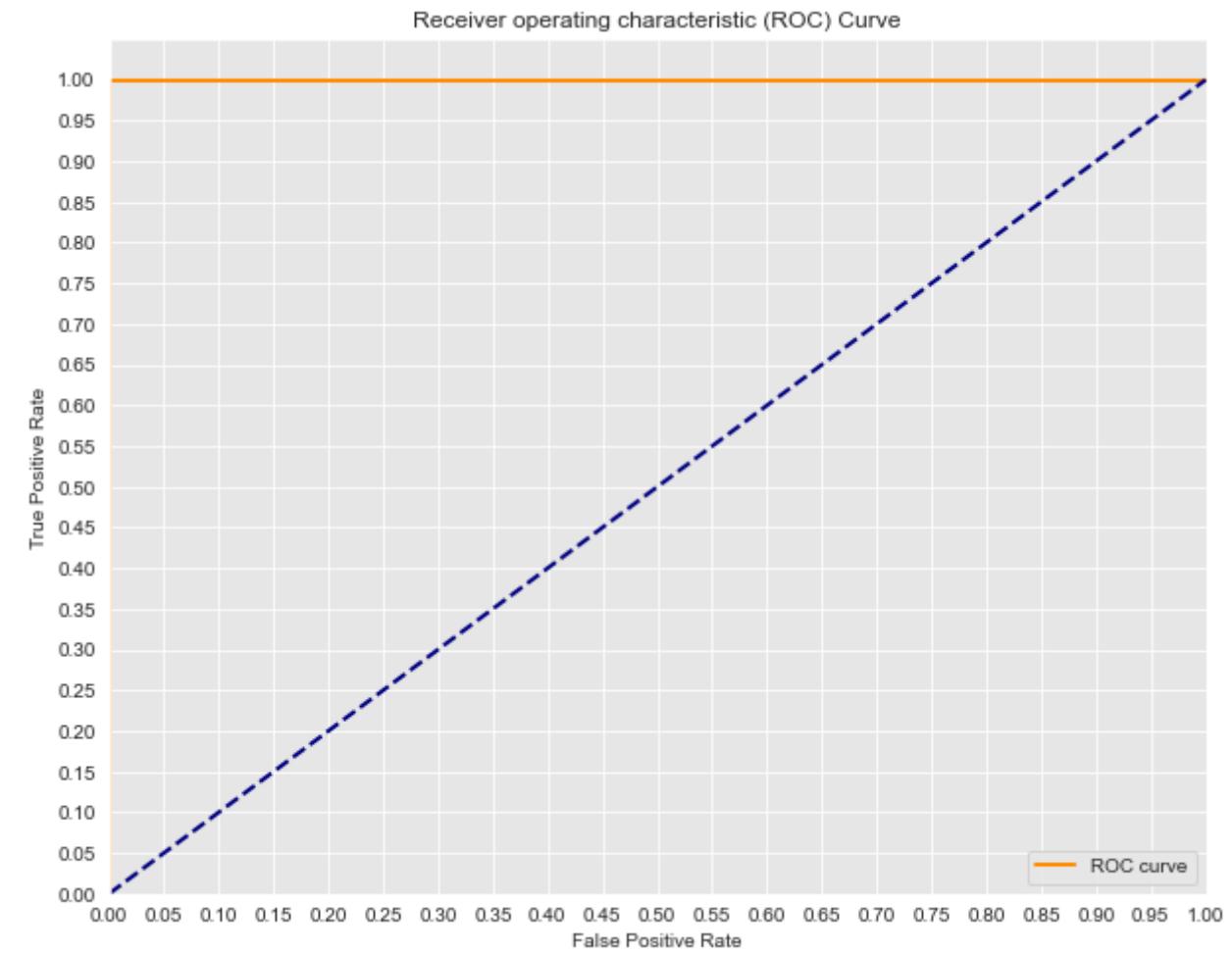




Cross Validated ROC AUC score: 1.0

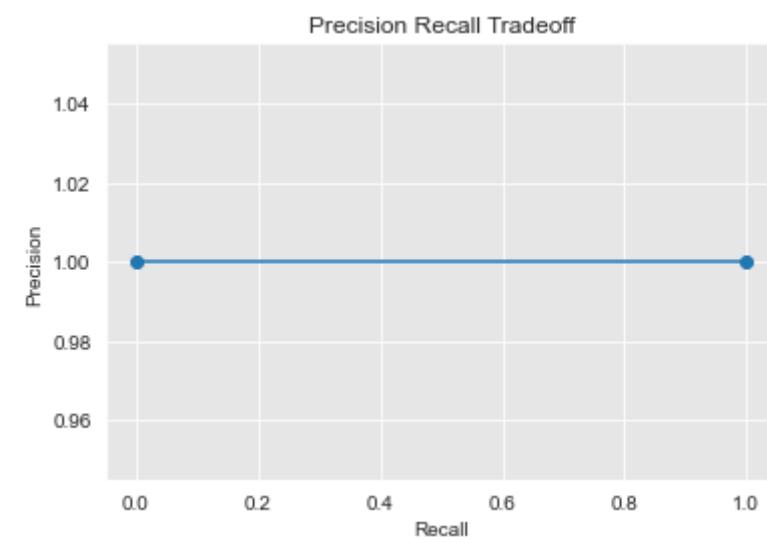
Class:N
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

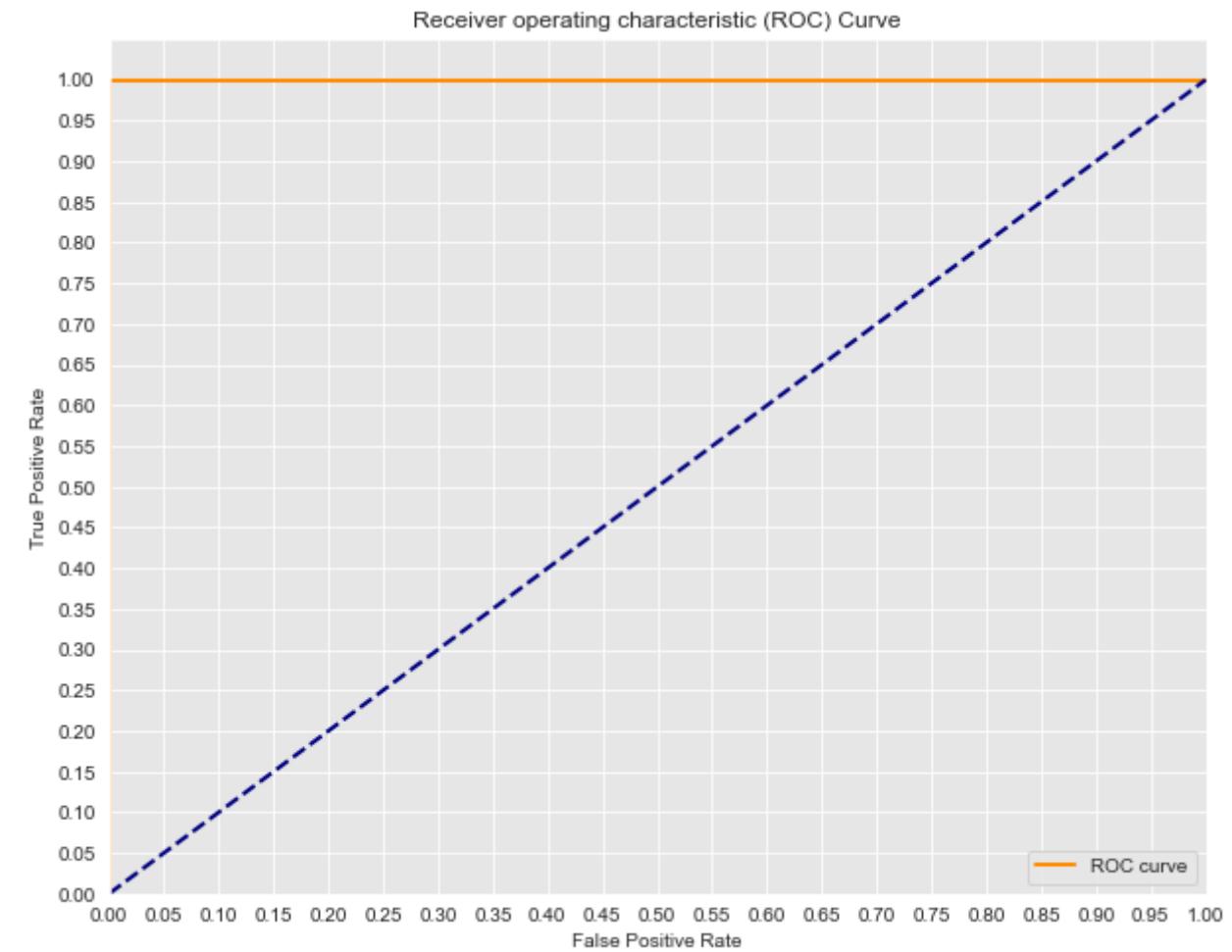




Cross Validated ROC AUC score: 1.0

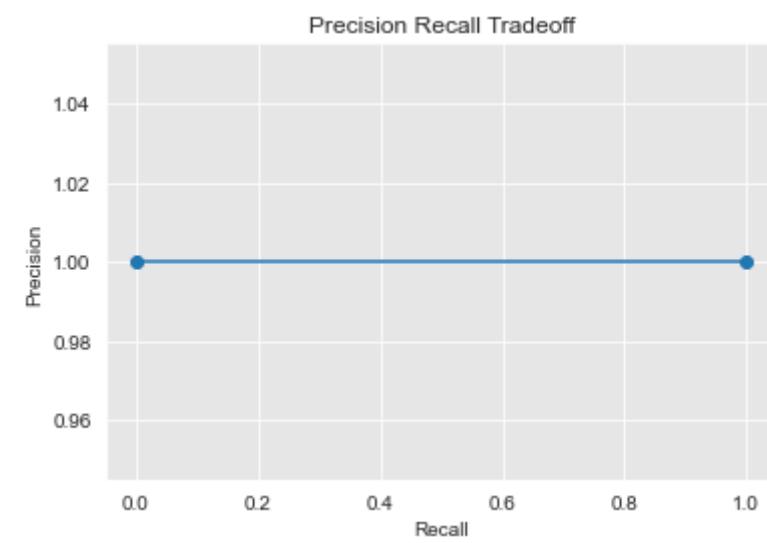
Class:P
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

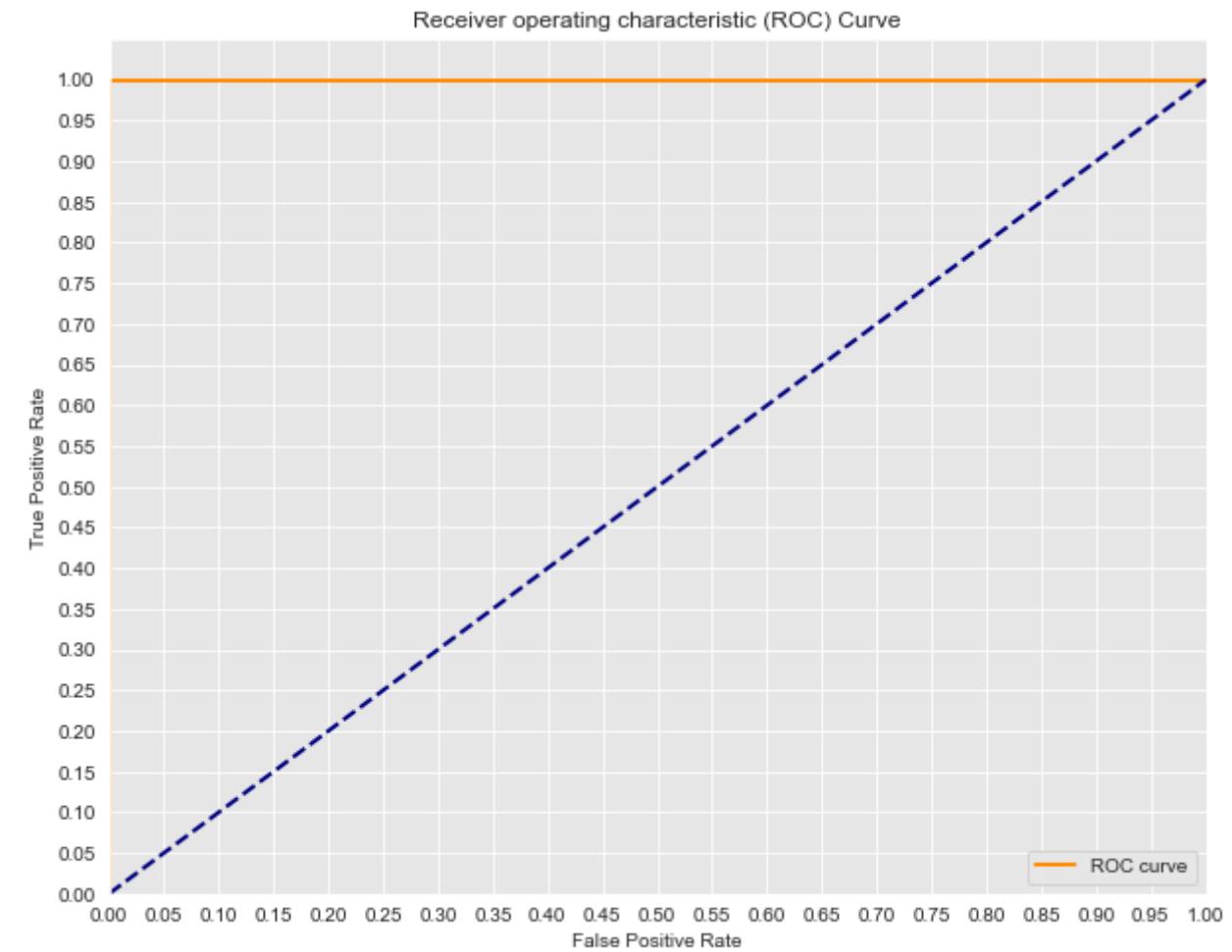




Cross Validated ROC AUC score: 1.0

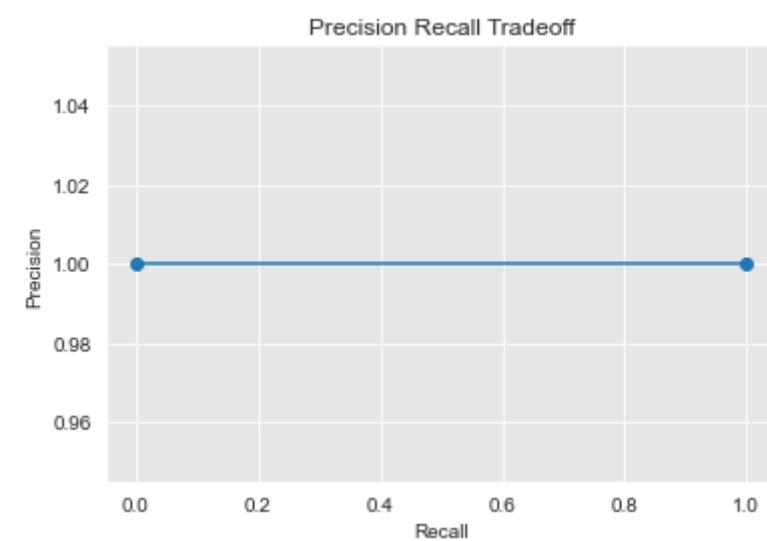
Class:Q
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

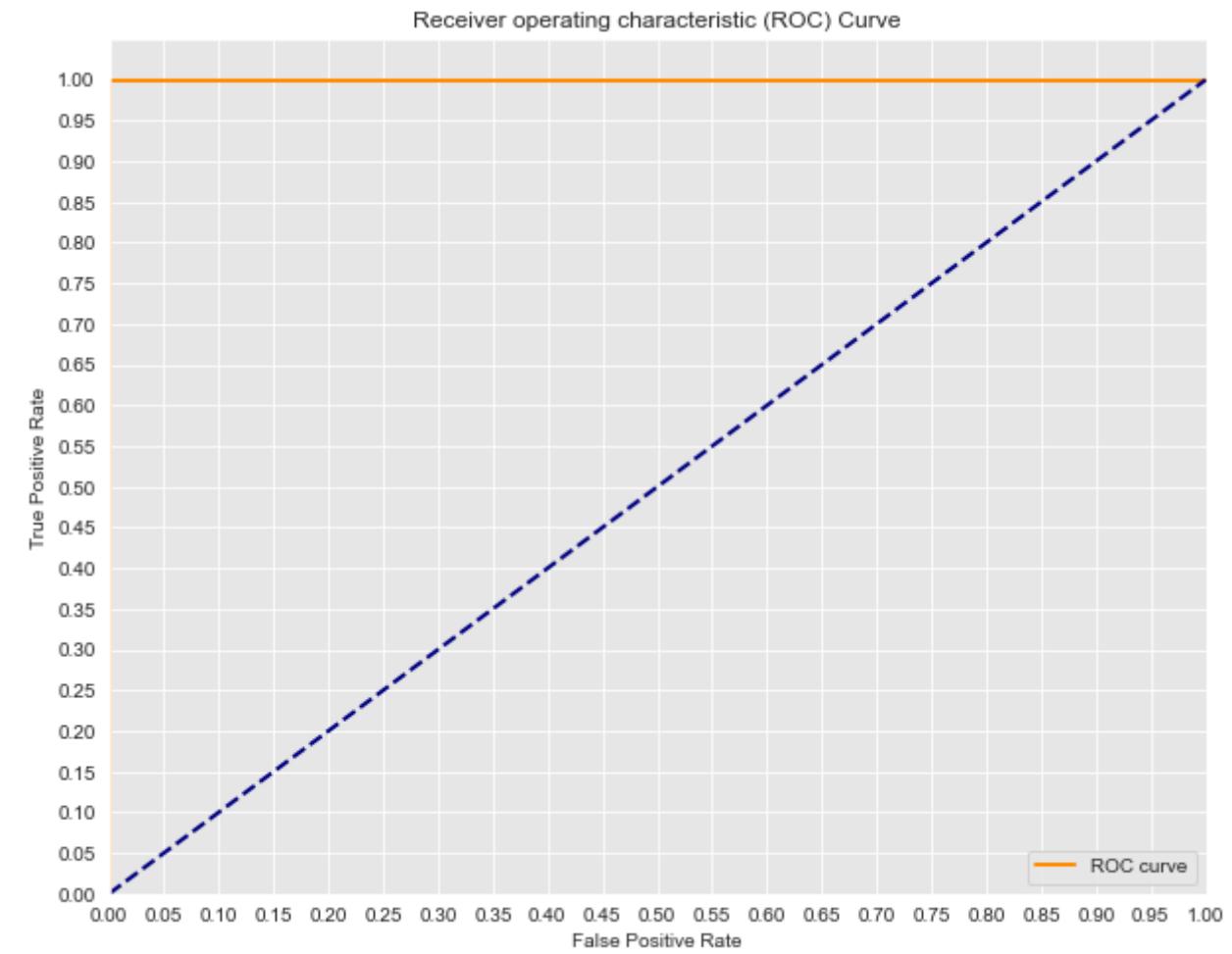




Cross Validated ROC AUC score: 1.0

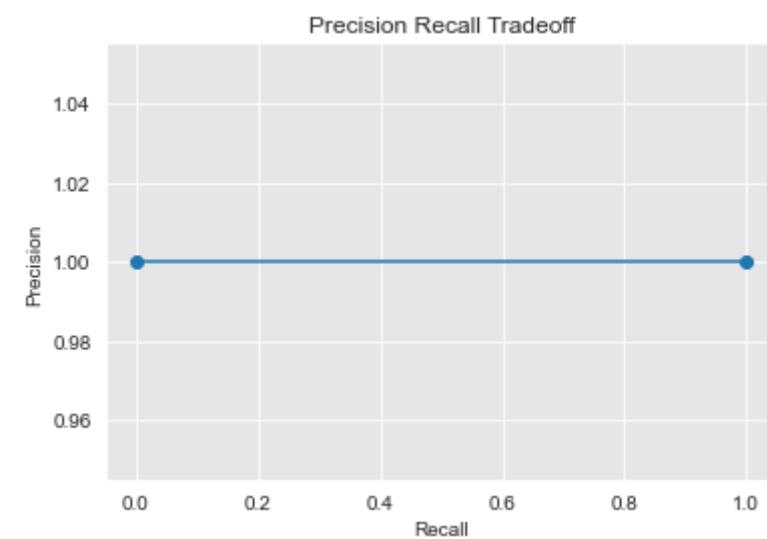
Class:R
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

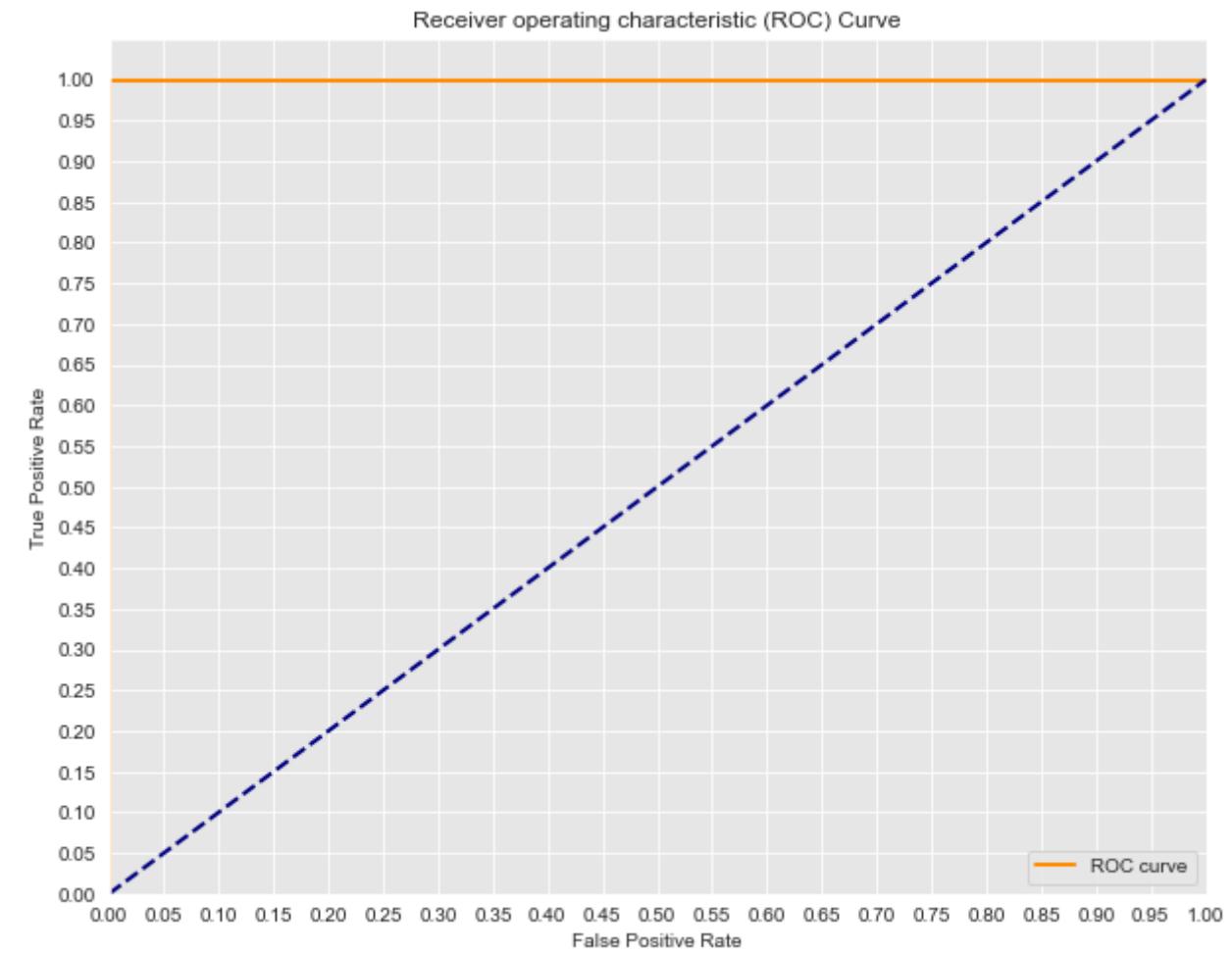




Cross Validated ROC AUC score: 1.0

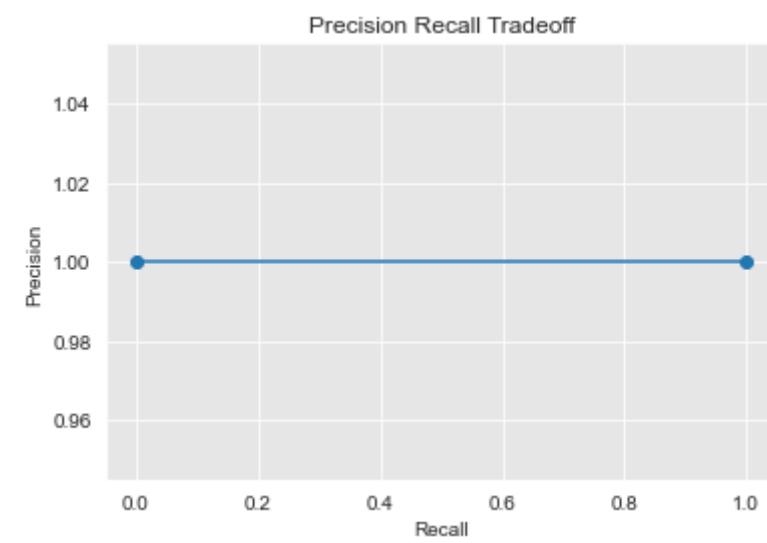
Class:S
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

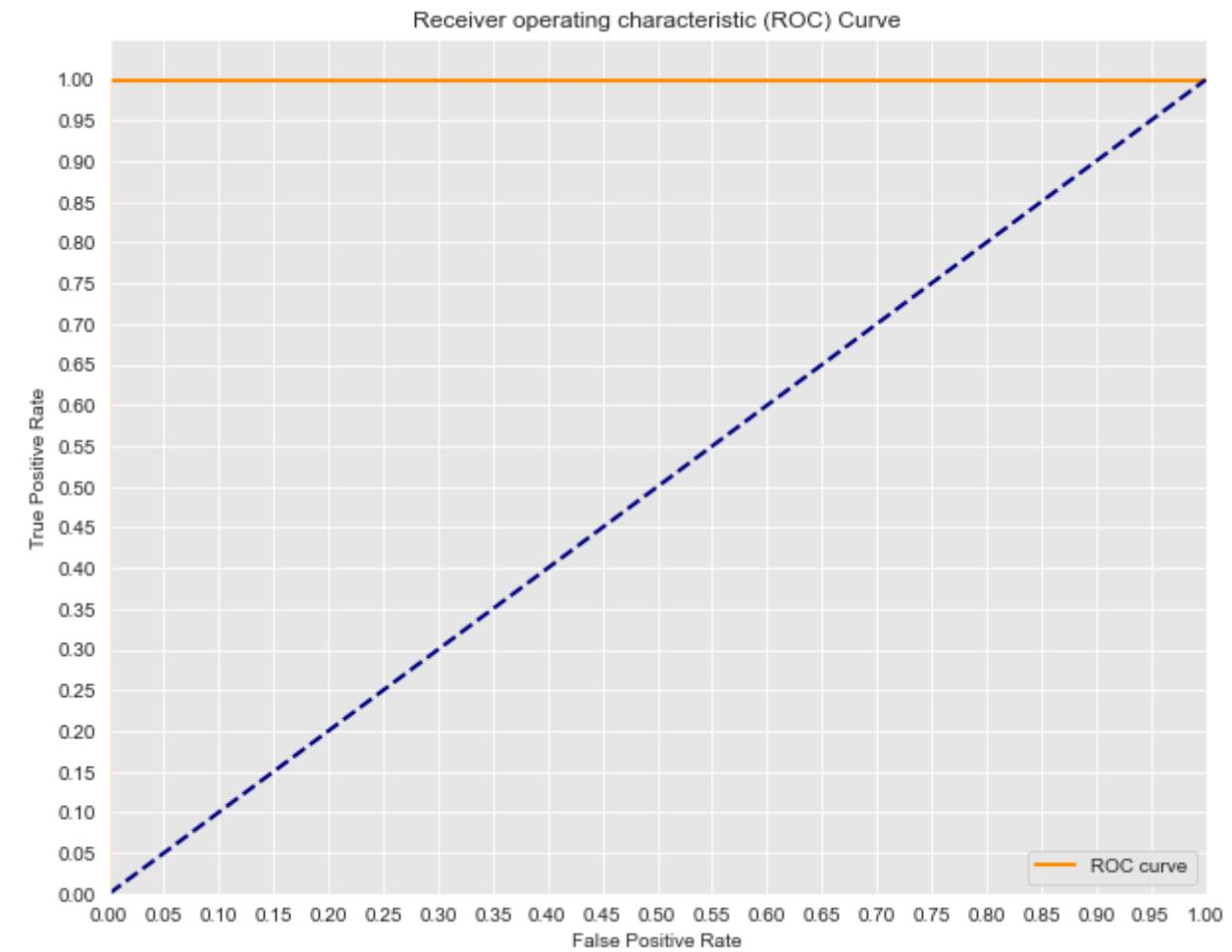




Cross Validated ROC AUC score: 1.0

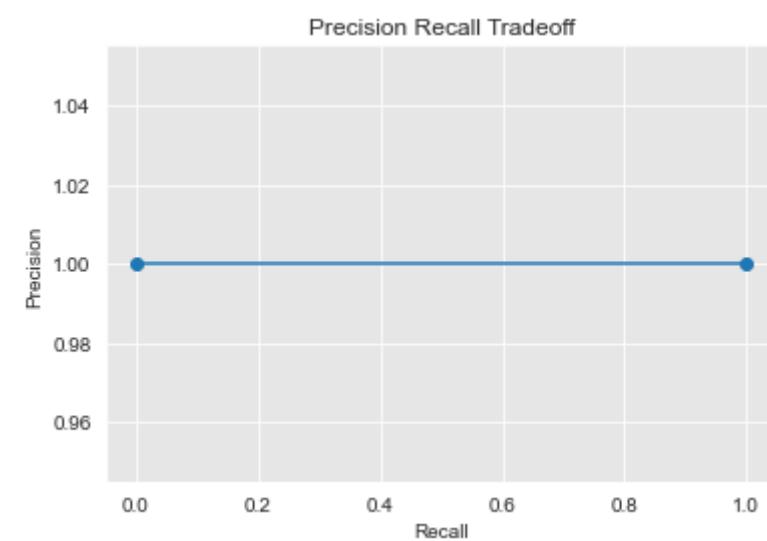
Class:Stop
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

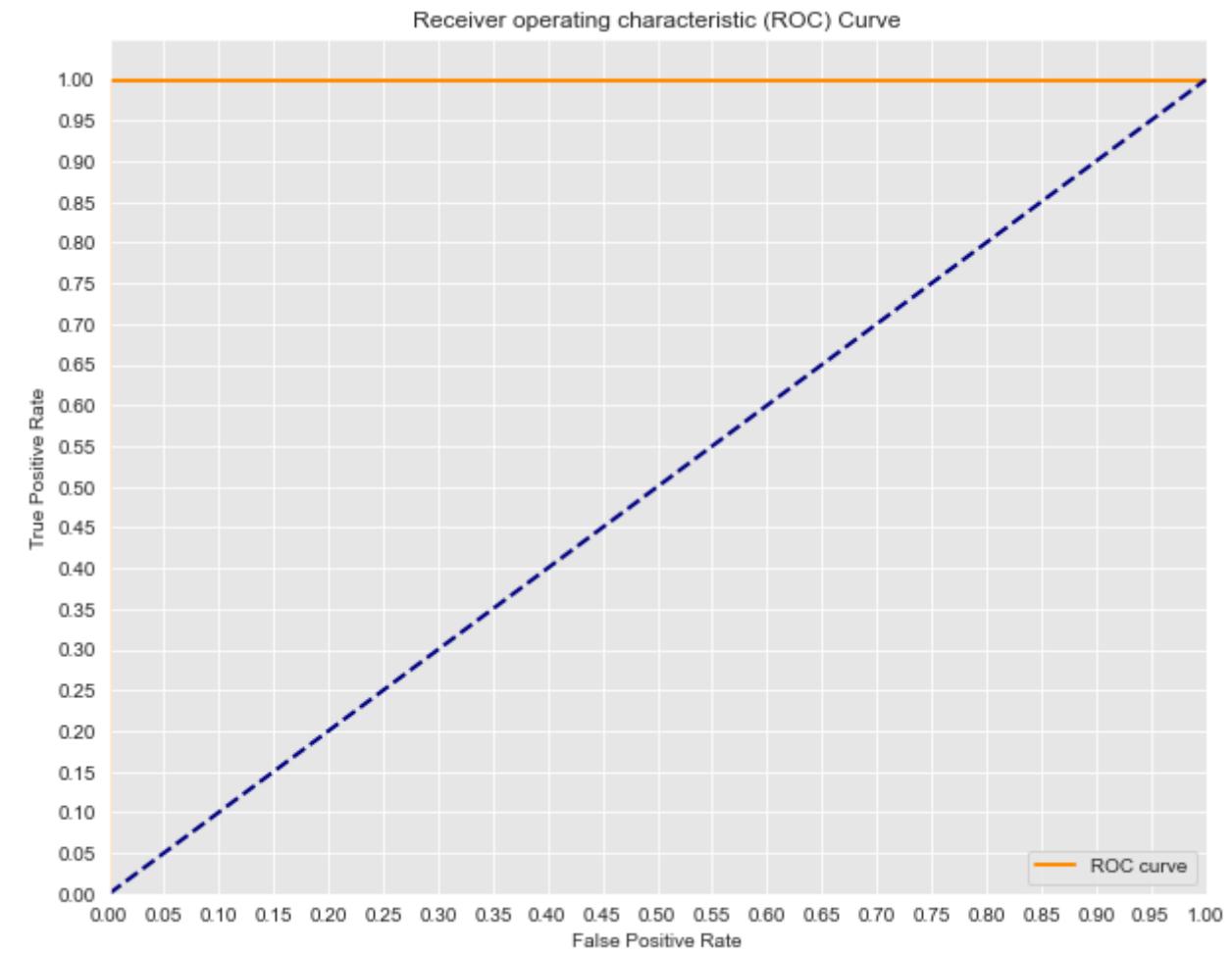




Cross Validated ROC AUC score: 1.0

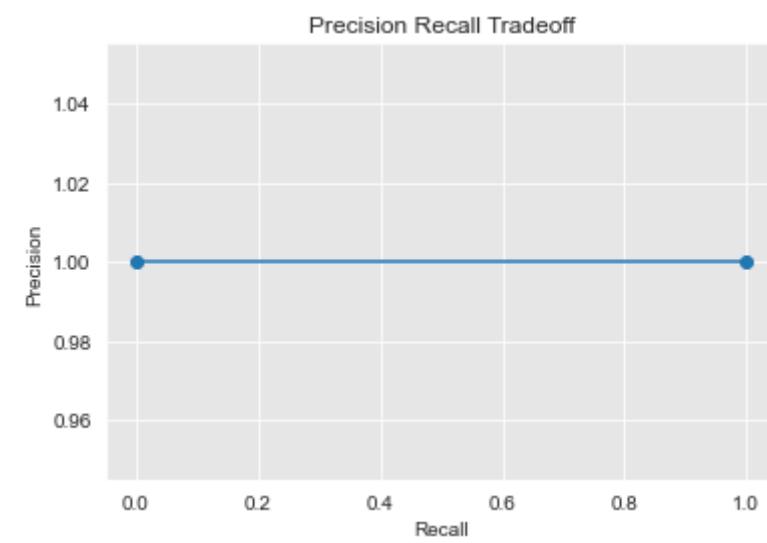
Class:T
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

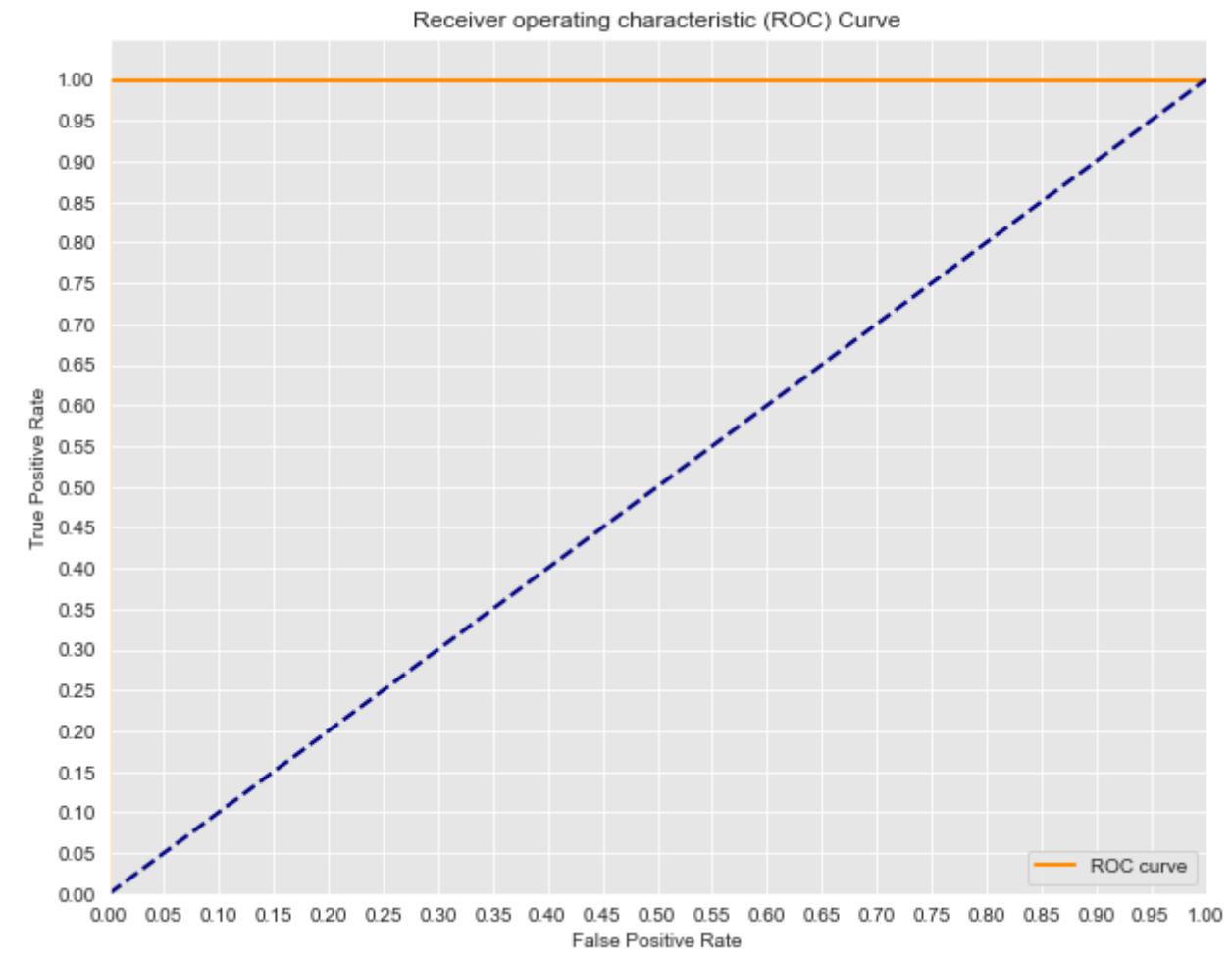




Cross Validated ROC AUC score: 1.0

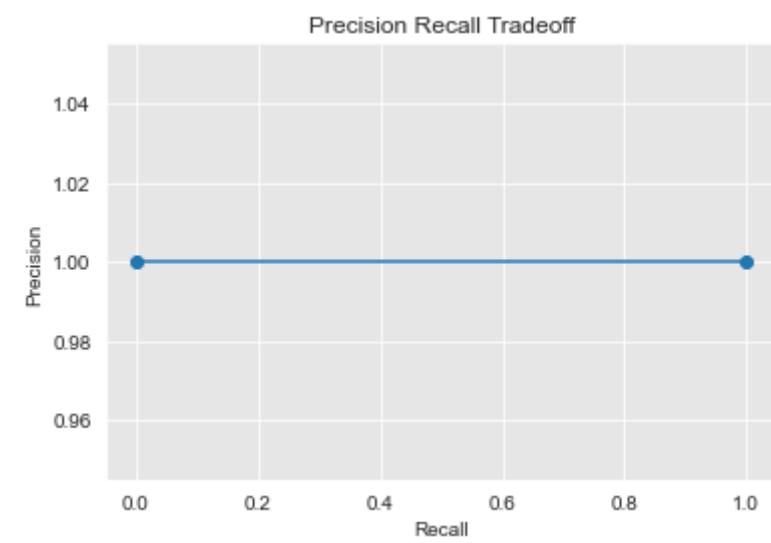
Class:V
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

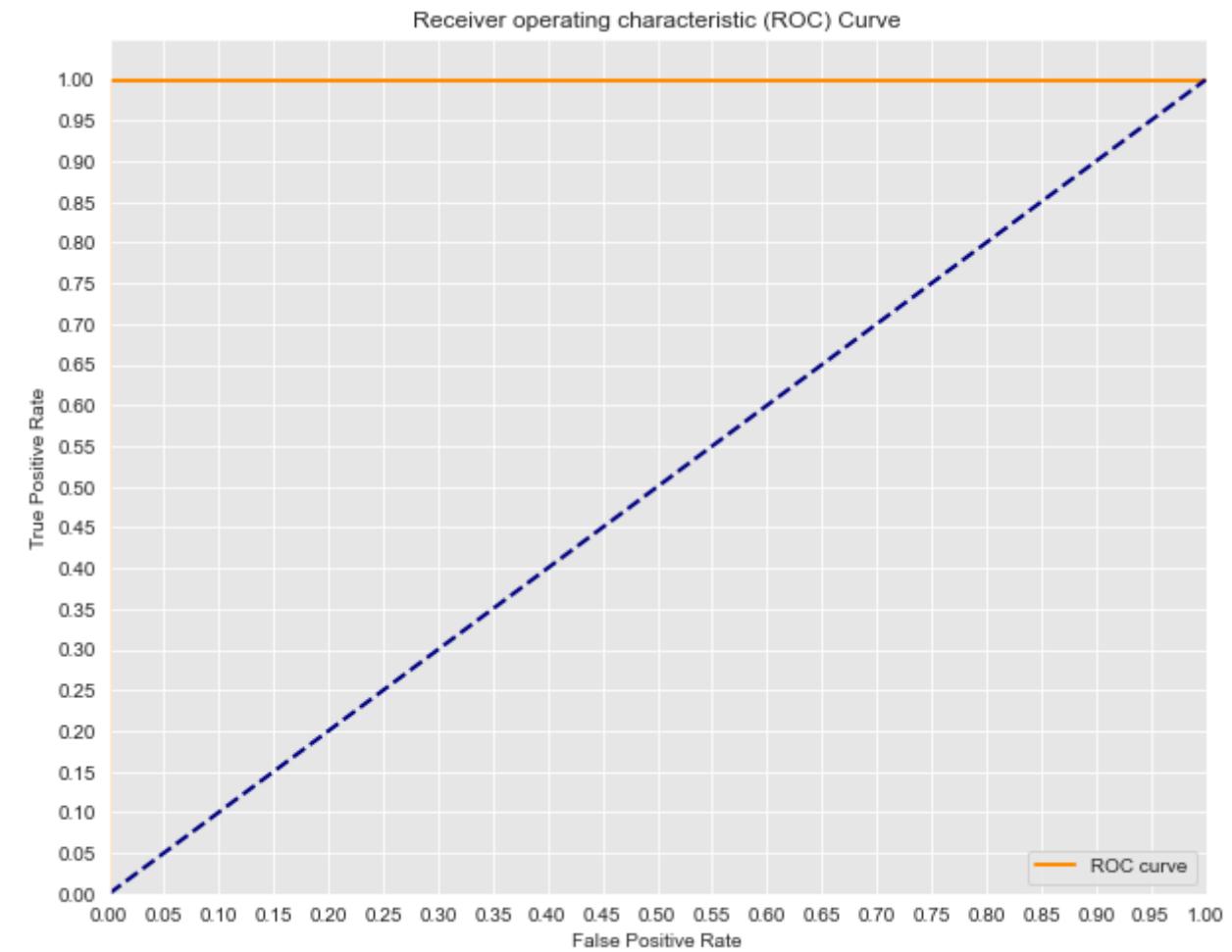




Cross Validated ROC AUC score: 1.0

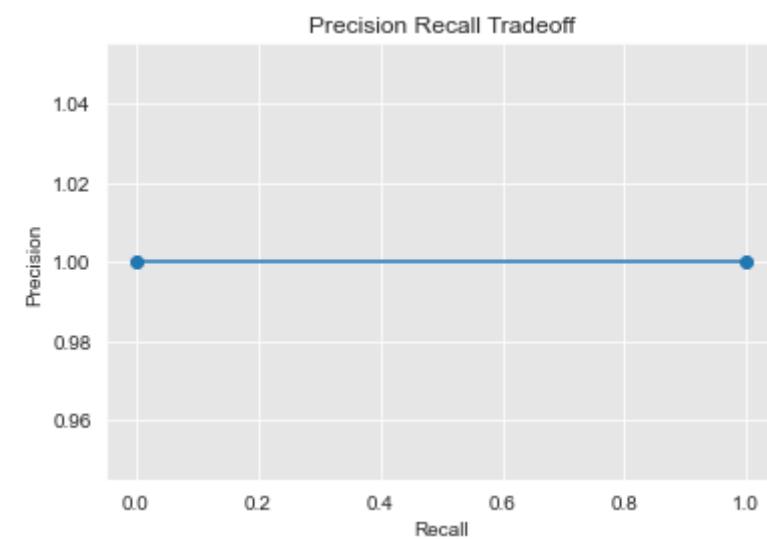
Class:W
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

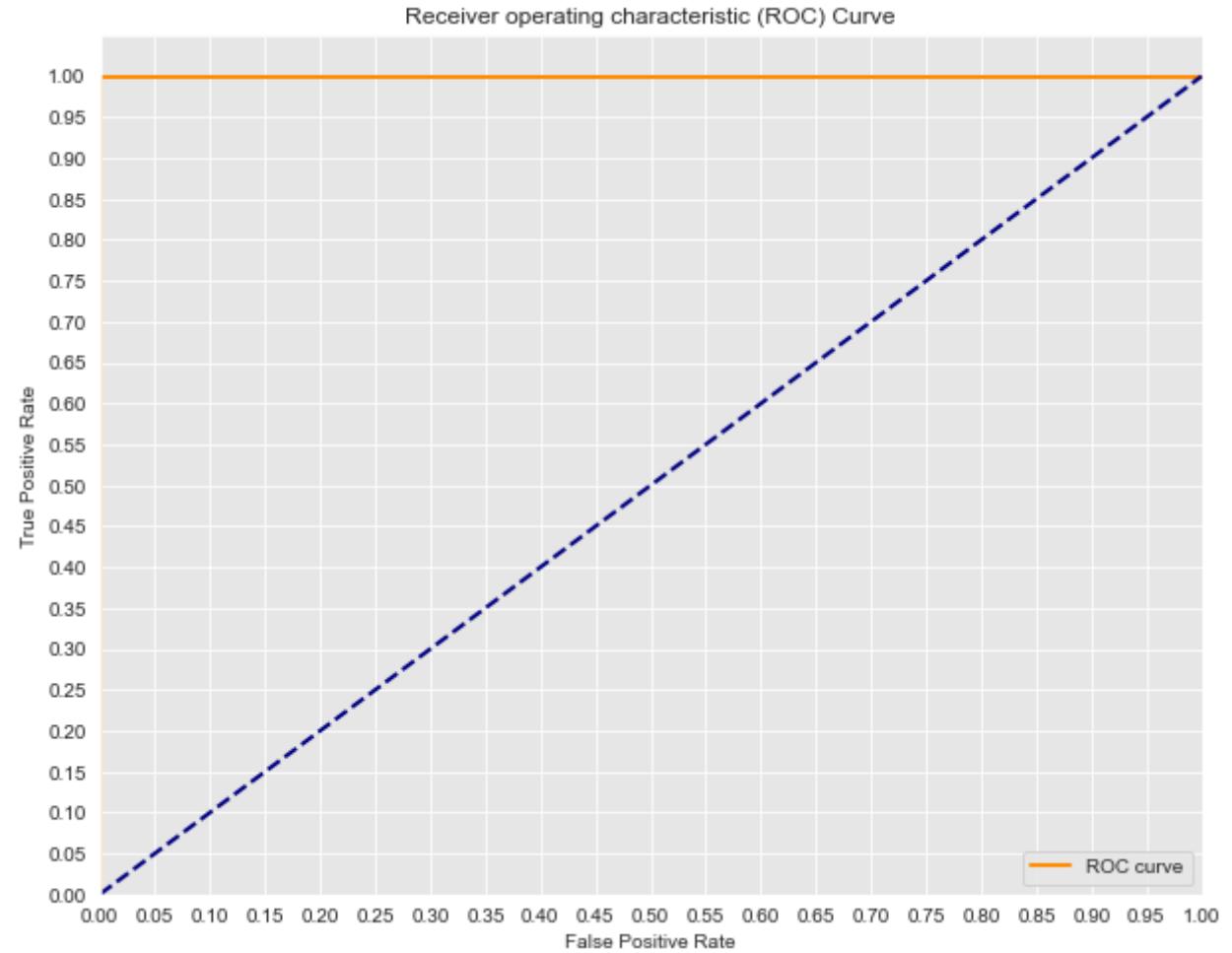




Cross Validated ROC AUC score: 1.0

Class:Y
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

Fit the model to the training data.

```
In [137]: gradient_boost.fit(x2_train, y2_train)
```

```
Out[137]: GradientBoostingClassifier()
```

Make predictions.

```
In [138]: y2_hat_train_gb = gradient_boost.predict(x2_train)
```

All predictions were classified correctly.

```
In [139]: con_mat(y2_train,y2_hat_train_gb)
```

Confusion Matrix

	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	\
0	370	0	0	0	0	0	0	0	0	0	...	0	0	0	0	\
1	0	182	0	0	0	0	0	0	0	0	...	0	0	0	0	\
2	0	0	273	0	0	0	0	0	0	0	...	0	0	0	0	\
3	0	0	0	447	0	0	0	0	0	0	...	0	0	0	0	\
4	0	0	0	0	833	0	0	0	0	0	...	0	0	0	0	\
5	0	0	0	0	0	720	0	0	0	0	...	0	0	0	0	\
6	0	0	0	0	0	0	142	0	0	0	...	0	0	0	0	\
7	0	0	0	0	0	0	0	315	0	0	...	0	0	0	0	\
8	0	0	0	0	0	0	0	0	485	0	...	0	0	0	0	\
9	0	0	0	0	0	0	0	0	0	437	...	0	0	0	0	\
10	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	\

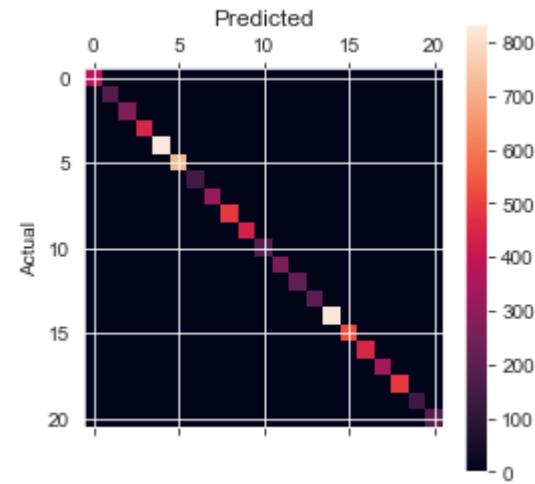
```

11   0   0   0   0   0   0   0   0   0   0   0   ... 259   0   0   0
12   0   0   0   0   0   0   0   0   0   0   0   ... 0   209   0   0
13   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   200   0
14   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   824
15   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
16   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
17   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
18   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
19   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0
20   0   0   0   0   0   0   0   0   0   0   0   ... 0   0   0   0

```

	15	16	17	18	19	20
0	0	0	0	0	0	0
1	0	0	0	0	0	0
2	0	0	0	0	0	0
3	0	0	0	0	0	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0
7	0	0	0	0	0	0
8	0	0	0	0	0	0
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	0	0	0	0	0	0
13	0	0	0	0	0	0
14	0	0	0	0	0	0
15	522	0	0	0	0	0
16	0	451	0	0	0	0
17	0	0	339	0	0	0
18	0	0	0	488	0	0
19	0	0	0	0	134	0
20	0	0	0	0	0	183

[21 rows x 21 columns]



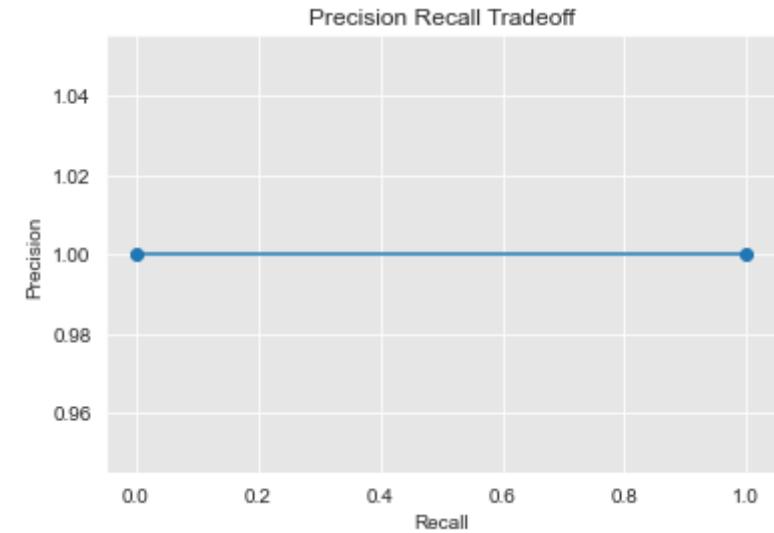
Separate predicted values into each class for metrics.

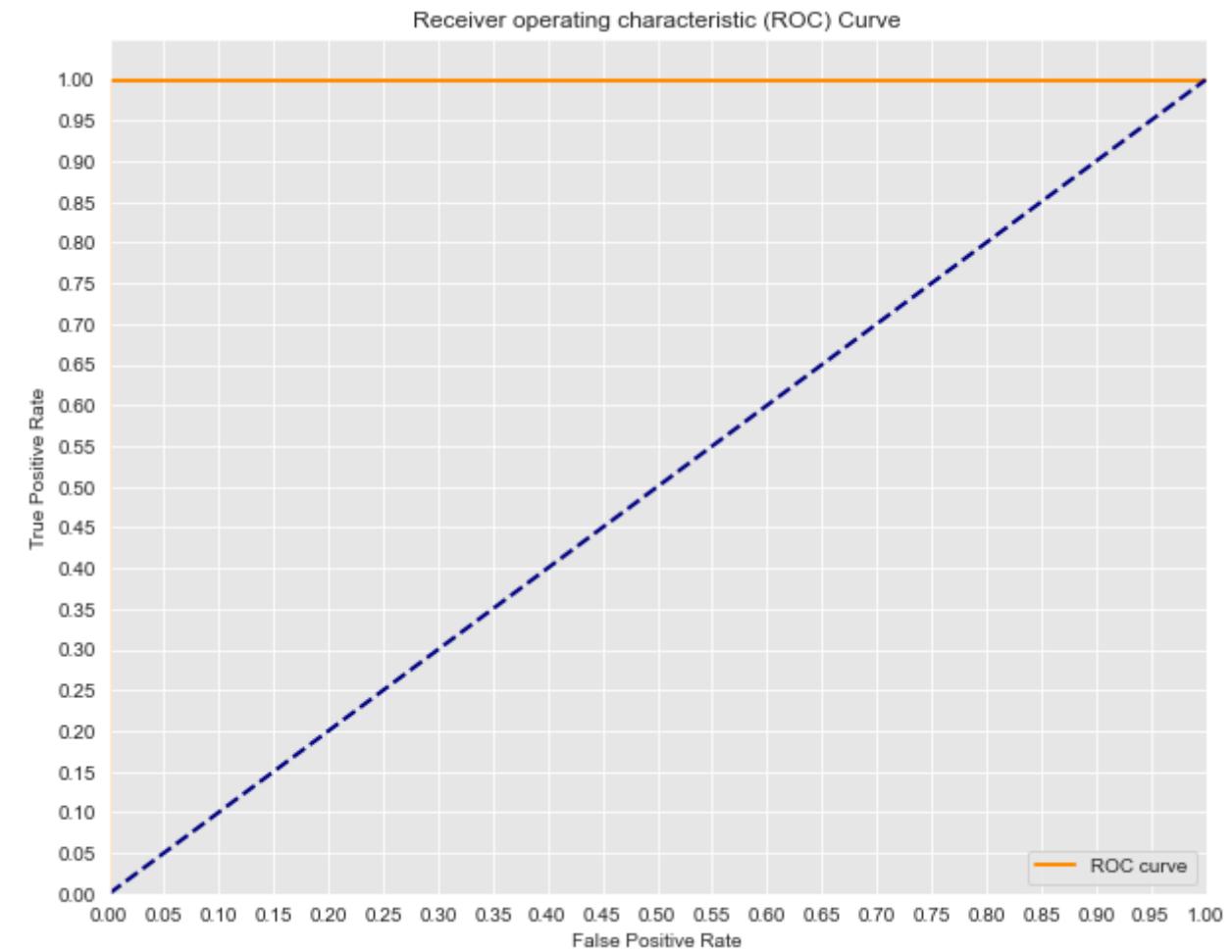
```
In [141]: predictions_class0=y2_hat_train_gb==0
predictions_class1=y2_hat_train_gb==1
predictions_class2=y2_hat_train_gb==2
predictions_class3=y2_hat_train_gb==3
predictions_class4=y2_hat_train_gb==4
predictions_class5=y2_hat_train_gb==5
predictions_class6=y2_hat_train_gb==6
predictions_class7=y2_hat_train_gb==7
predictions_class8=y2_hat_train_gb==8
```

```
predictions_class9=y2_hat_train_gb==9
predictions_class10=y2_hat_train_gb==10
predictions_class11=y2_hat_train_gb==11
predictions_class12=y2_hat_train_gb==12
predictions_class13=y2_hat_train_gb==13
predictions_class14=y2_hat_train_gb==14
predictions_class15=y2_hat_train_gb==15
predictions_class16=y2_hat_train_gb==16
predictions_class17=y2_hat_train_gb==17
predictions_class18=y2_hat_train_gb==18
predictions_class19=y2_hat_train_gb==19
predictions_class20=y2_hat_train_gb==20
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3,predictions_class4,predictions_class5,predictions_class6
    ,predictions_class7,predictions_class8,predictions_class9,predictions_class10,predictions_class11,predictions_class12
    ,predictions_class13,predictions_class14,predictions_class15,predictions_class16,predictions_class17,predictions_class18
    ,predictions_class19,predictions_class20]
```

```
In [142]: multiclass(gradient_boost,x2_train,a,p,translation_df2['Amino_Acids_copy'])
```

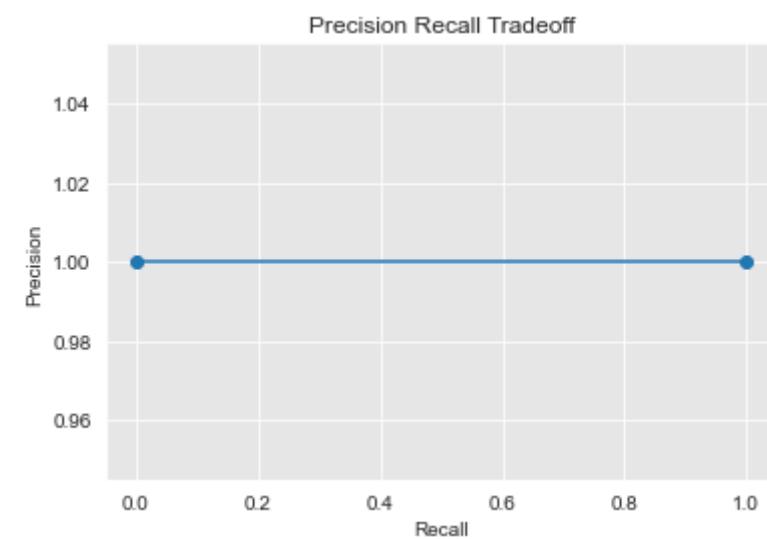
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

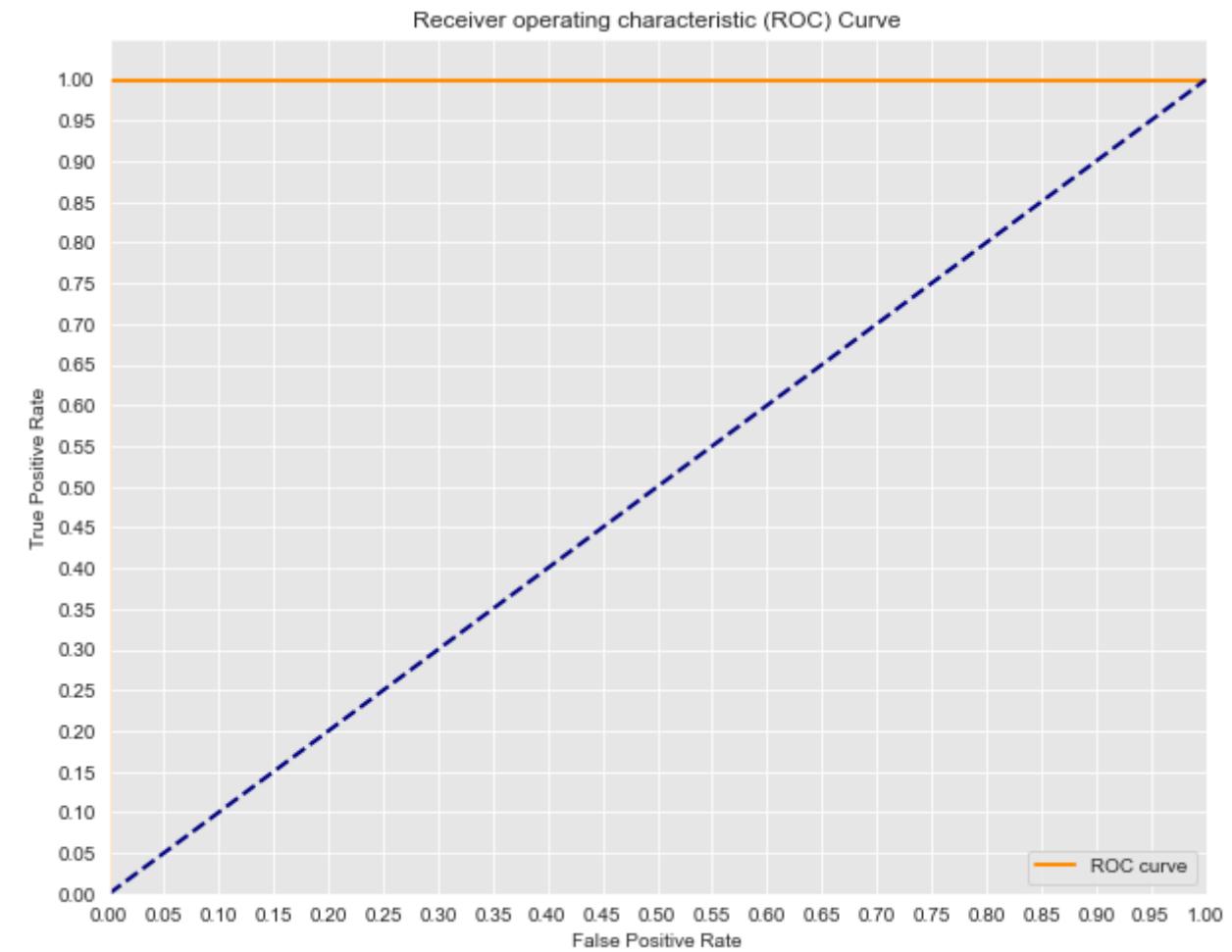




Cross Validated ROC AUC score: 1.0

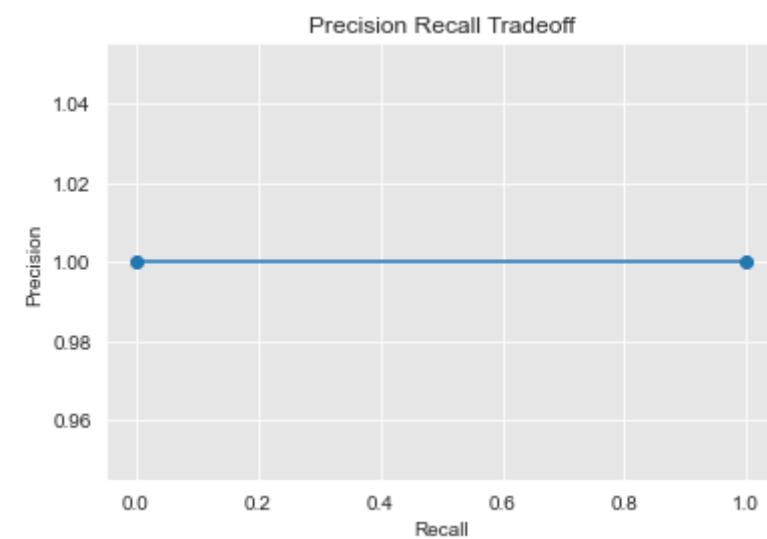
Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

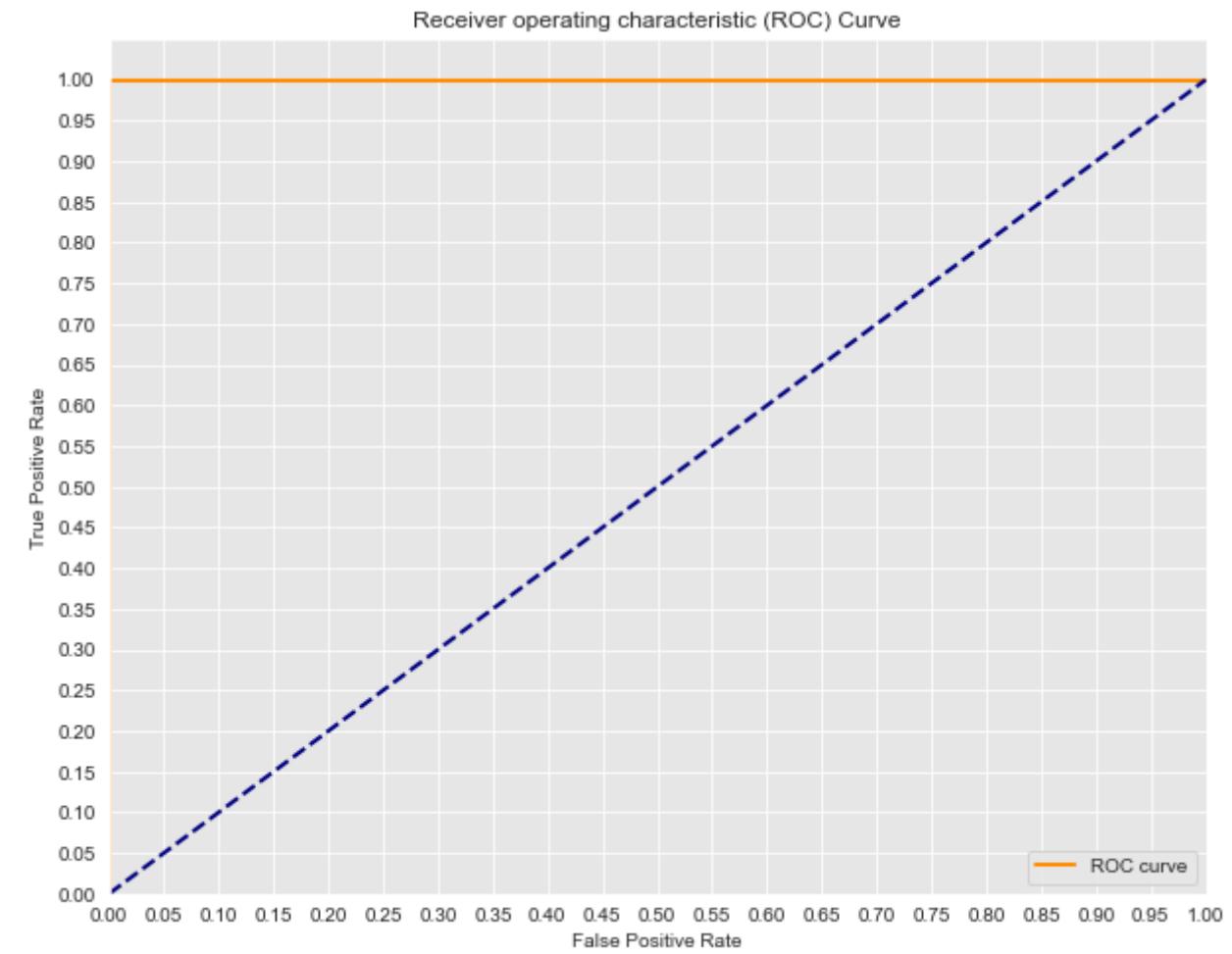




Cross Validated ROC AUC score: 1.0

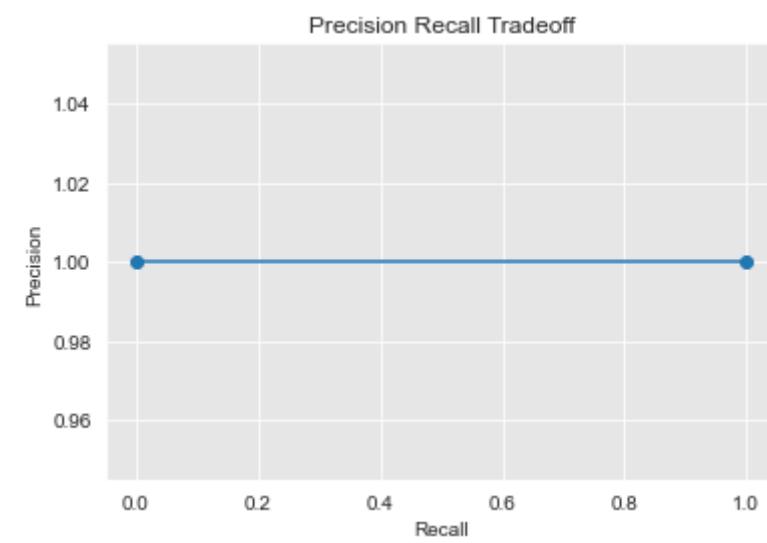
Class:D
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

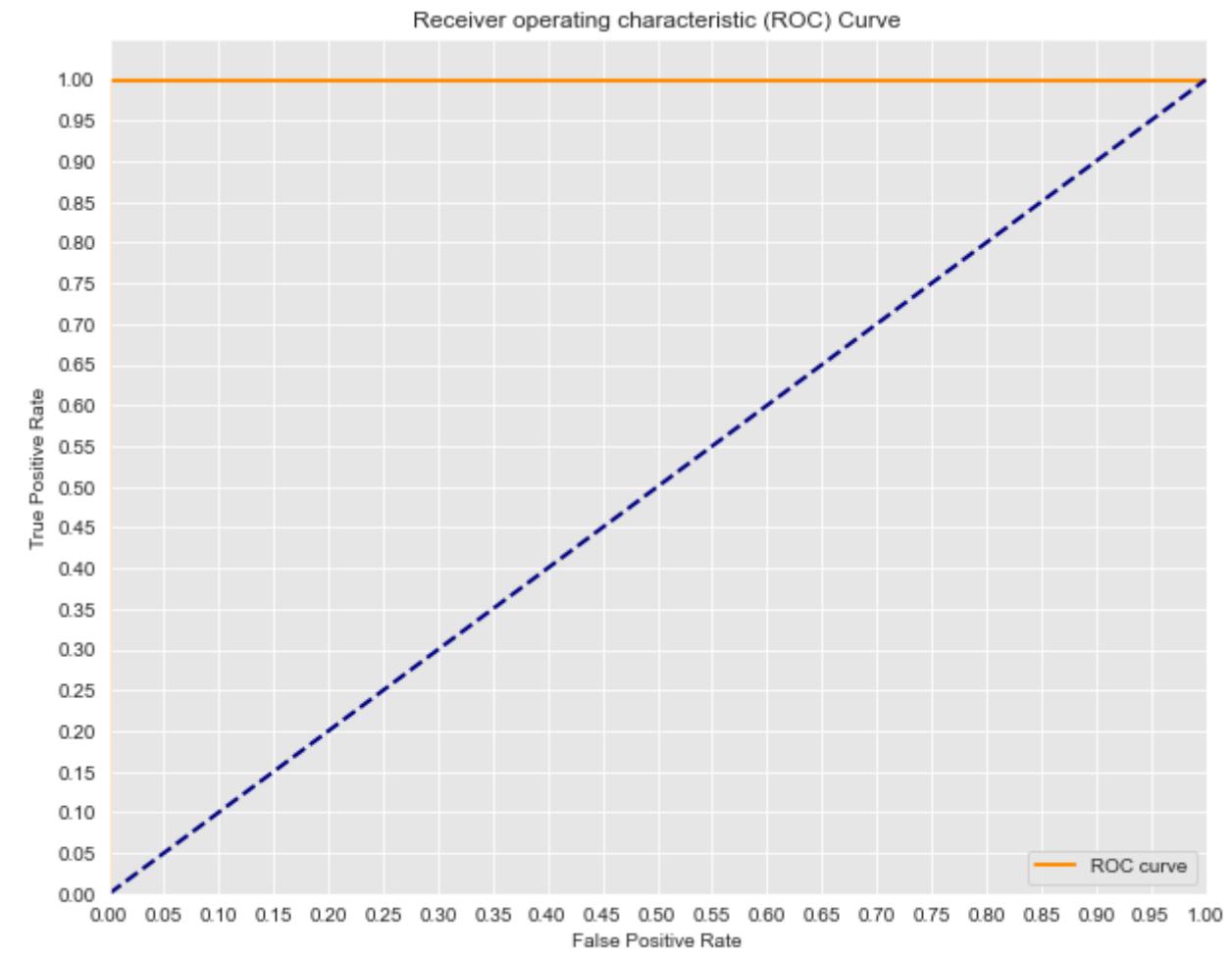




Cross Validated ROC AUC score: 1.0

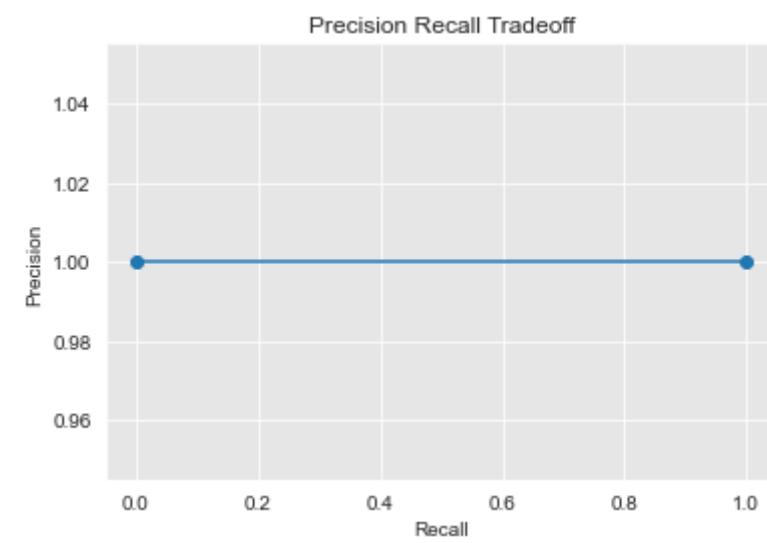
Class:E
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

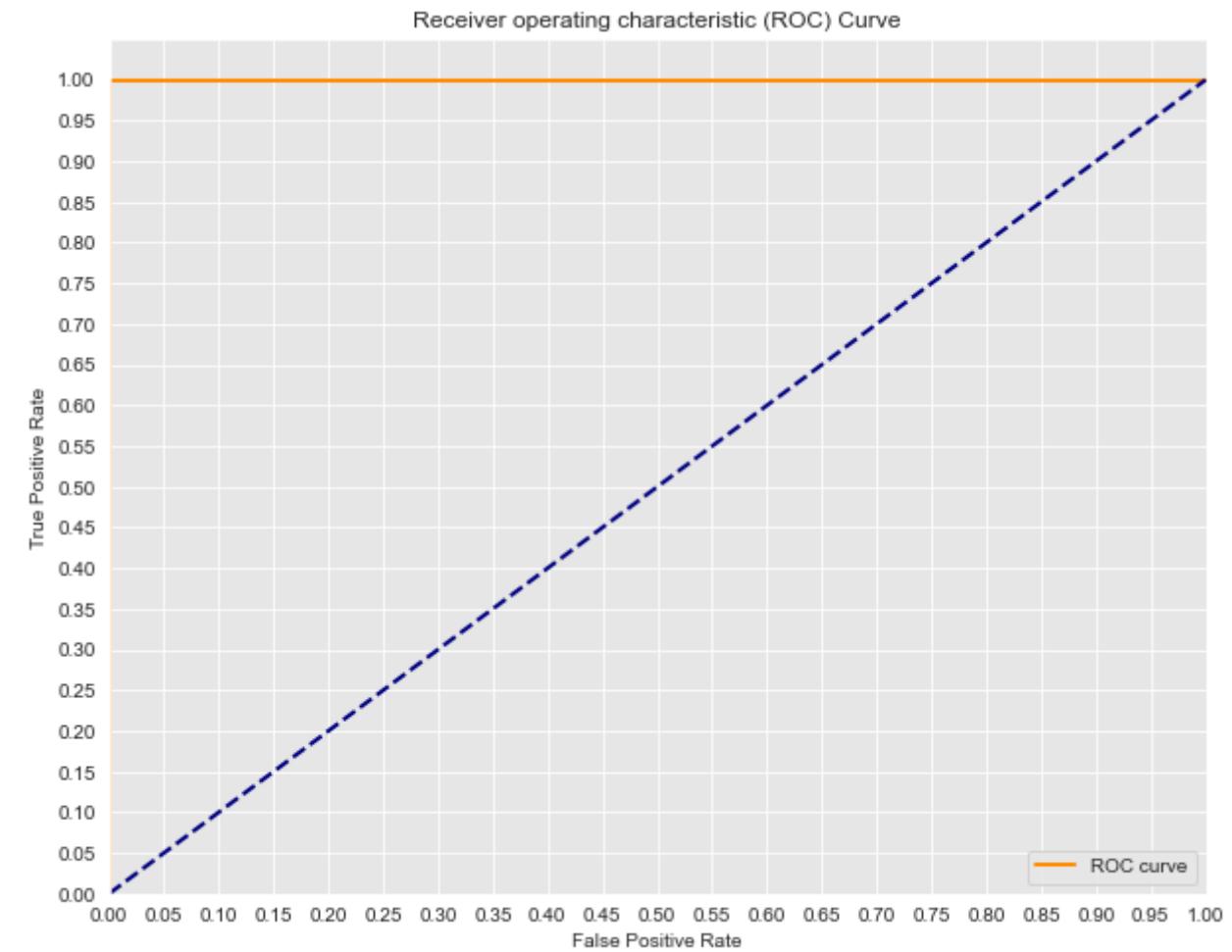




Cross Validated ROC AUC score: 1.0

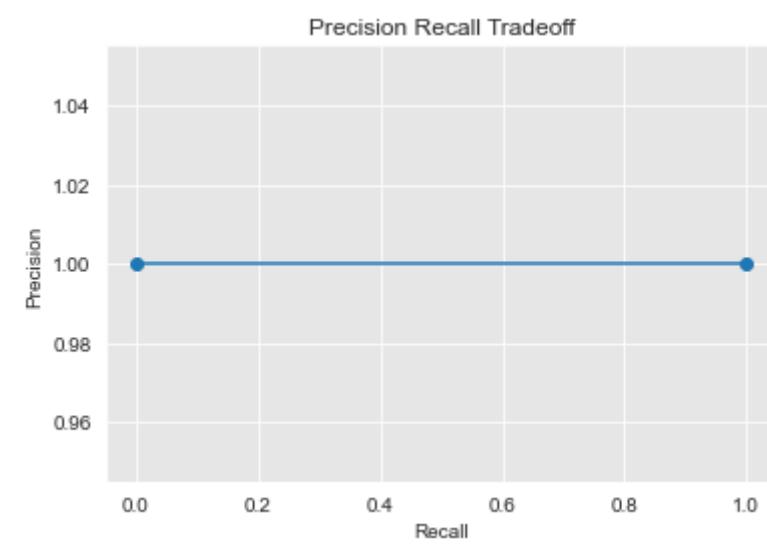
Class:F
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

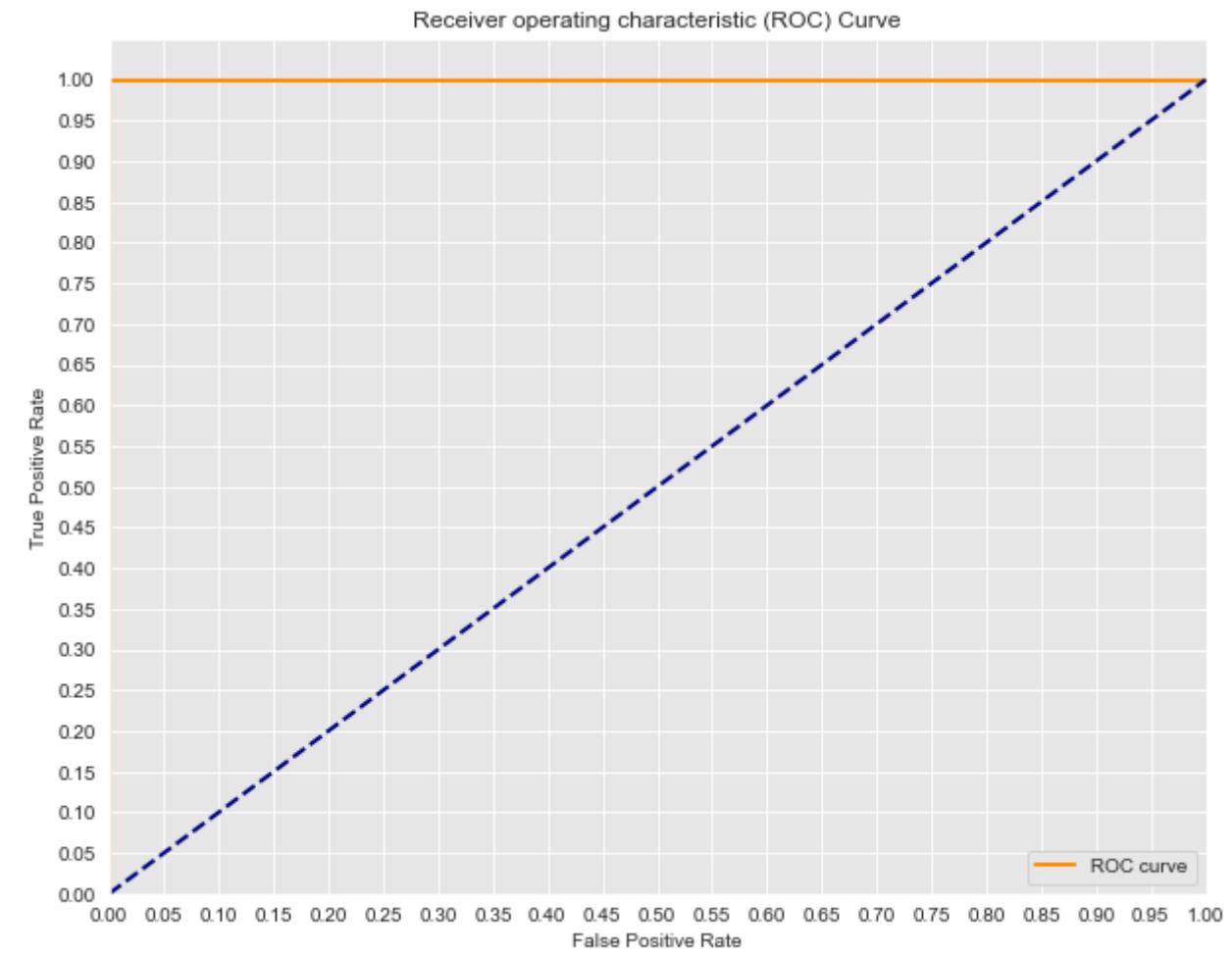




Cross Validated ROC AUC score: 1.0

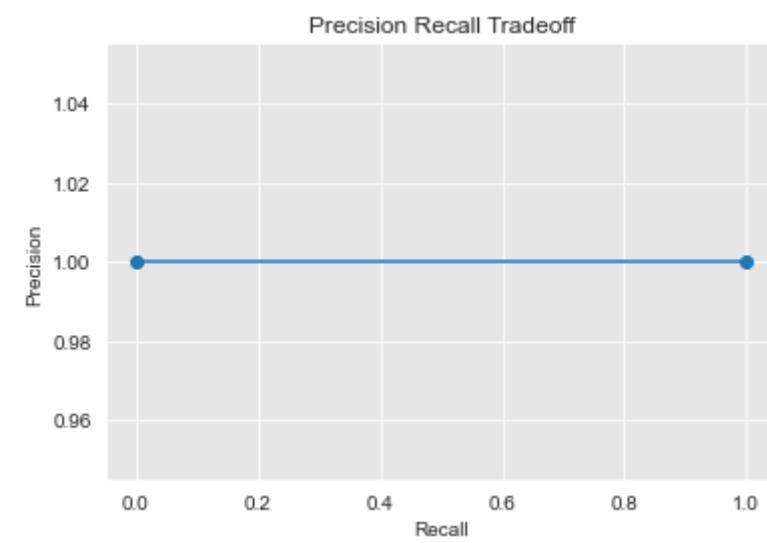
Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

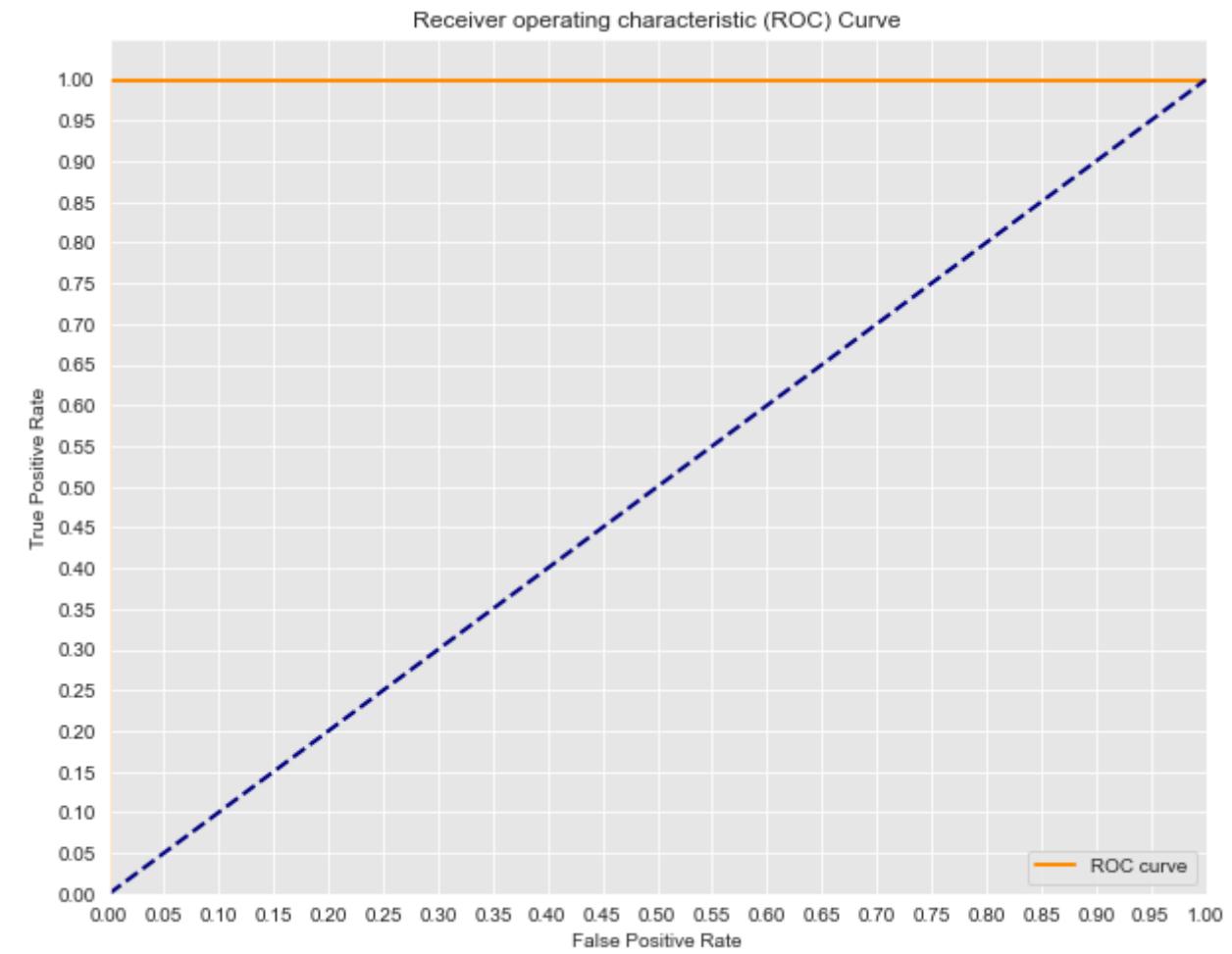




Cross Validated ROC AUC score: 1.0

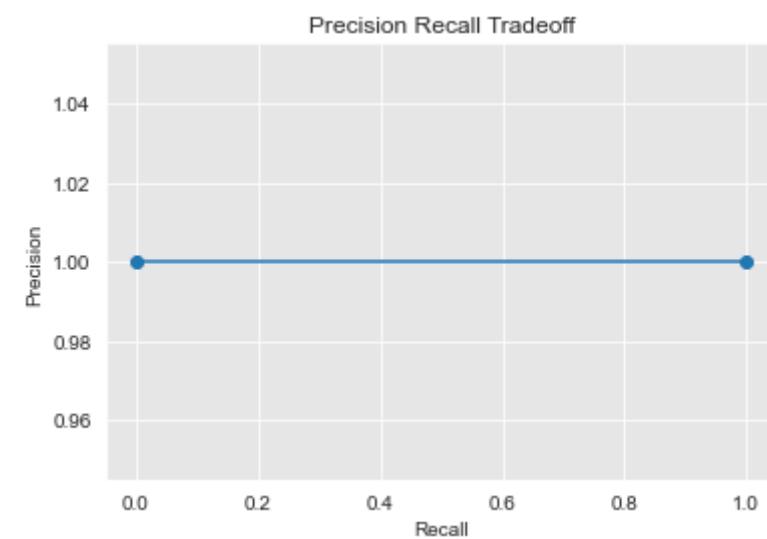
Class:H
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

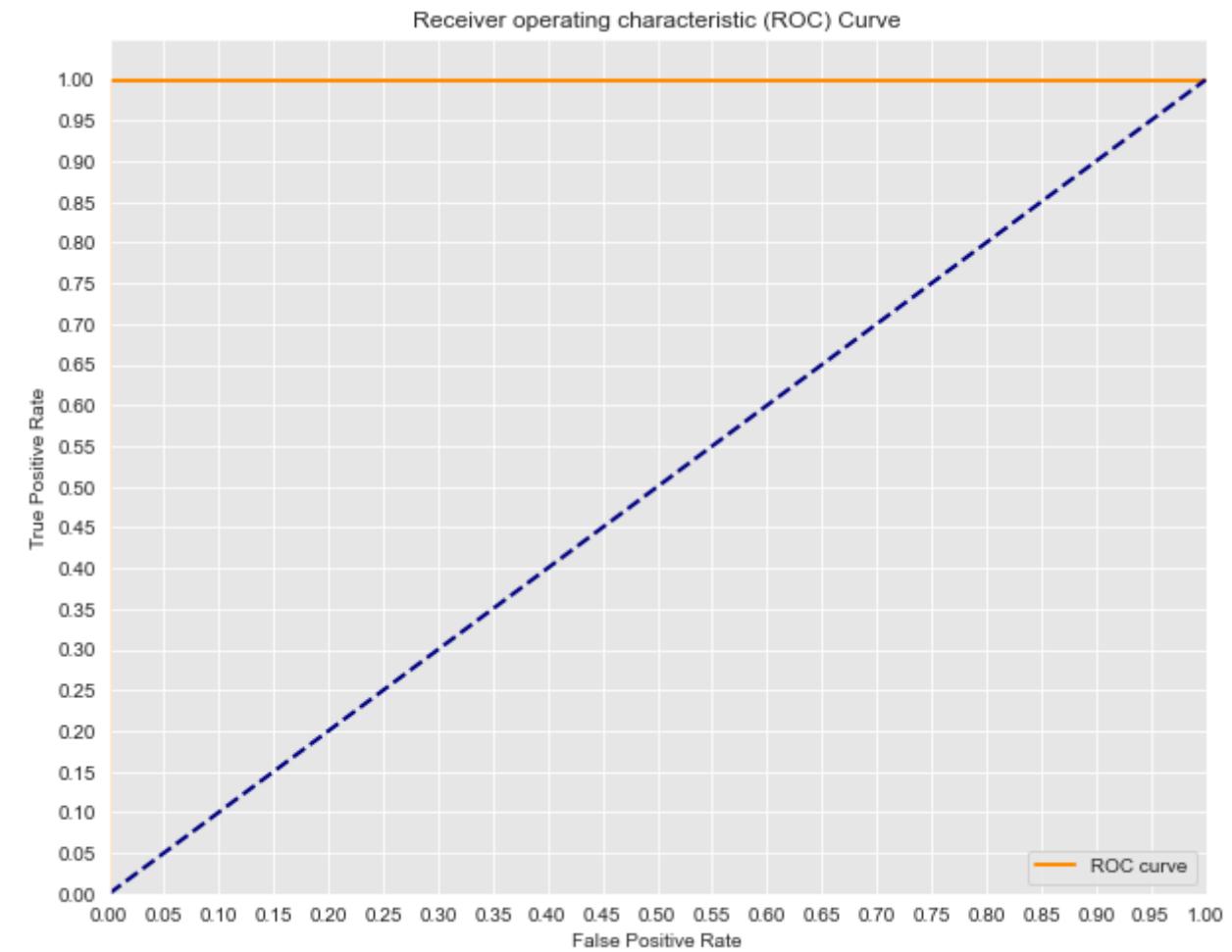




Cross Validated ROC AUC score: 1.0

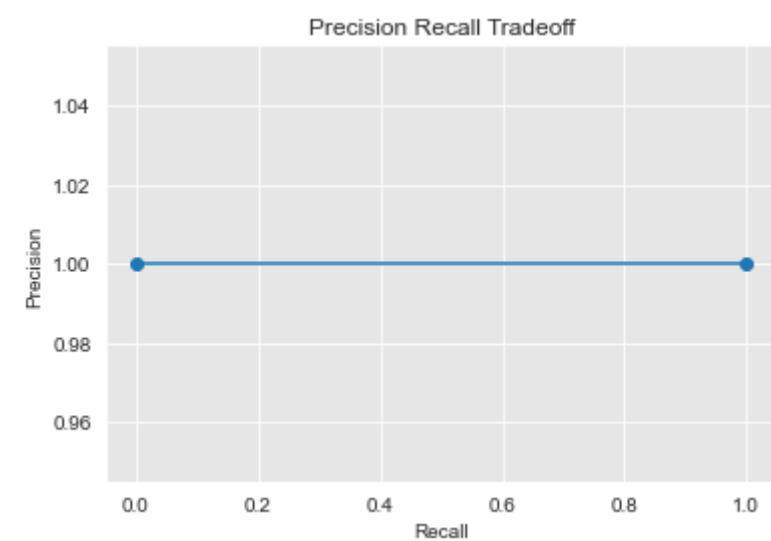
Class:I
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

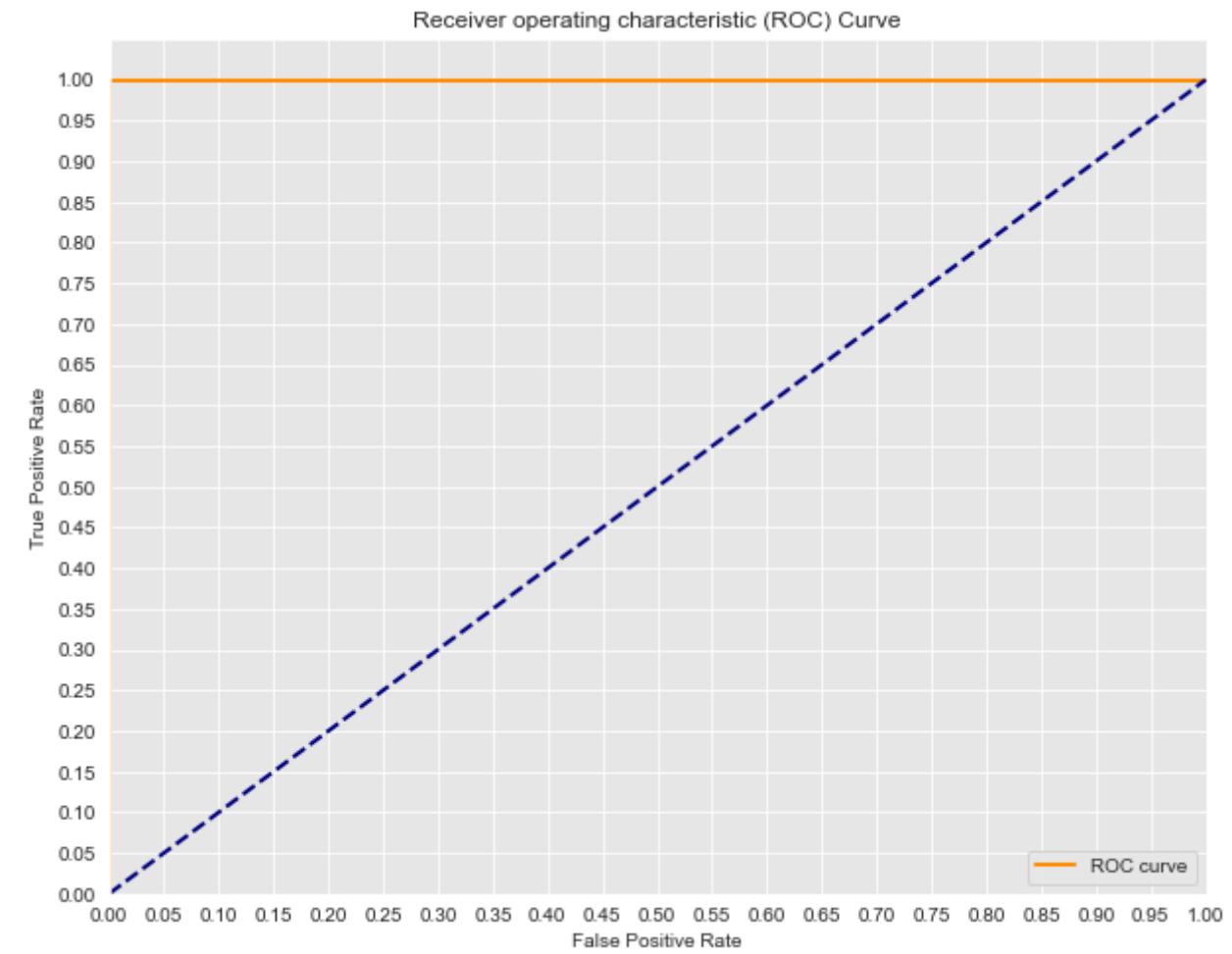




Cross Validated ROC AUC score: 1.0

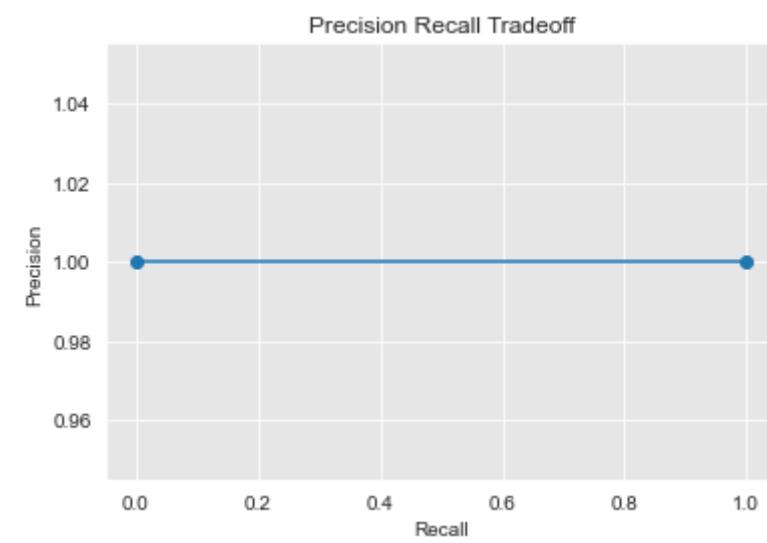
Class:K
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

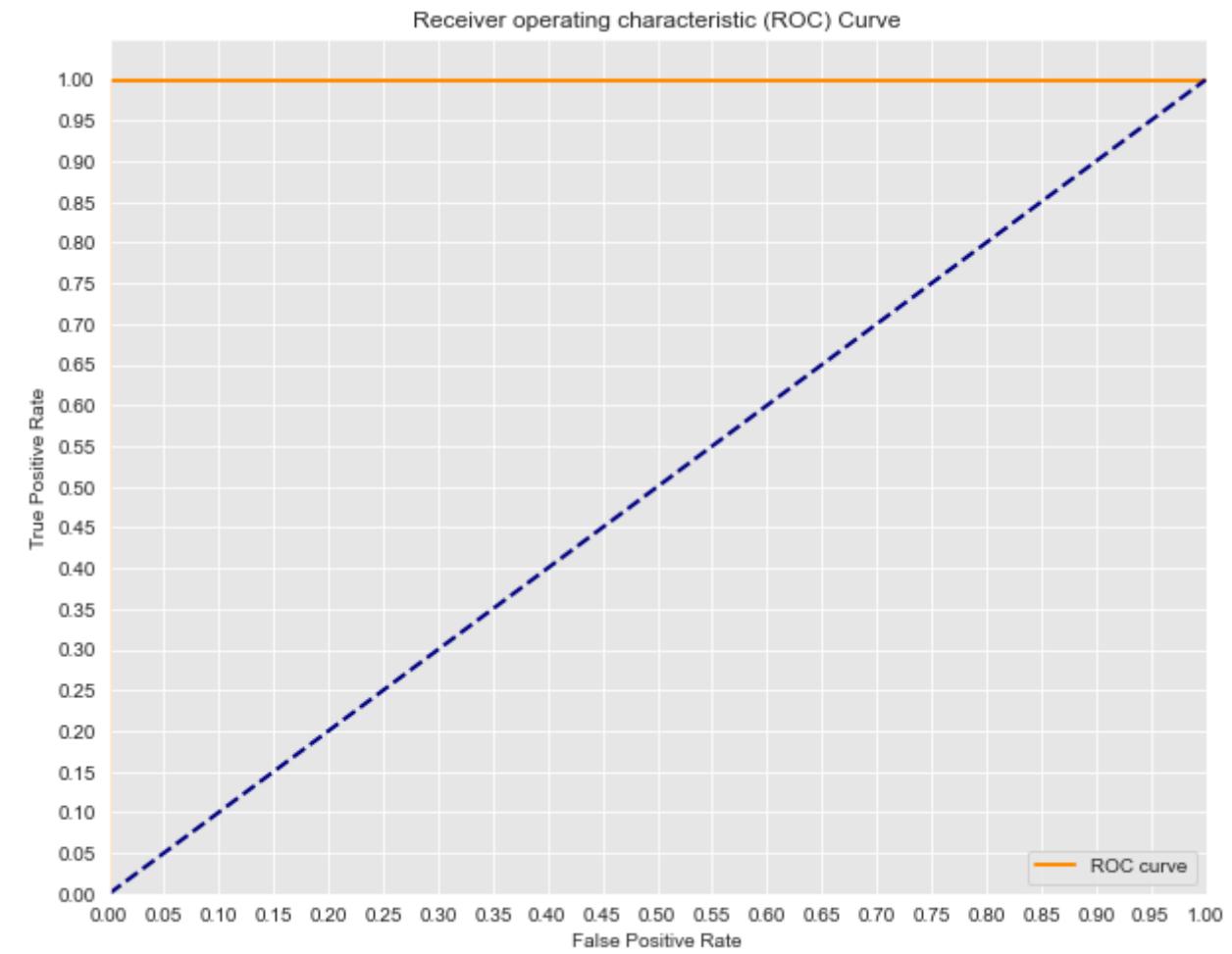




Cross Validated ROC AUC score: 1.0

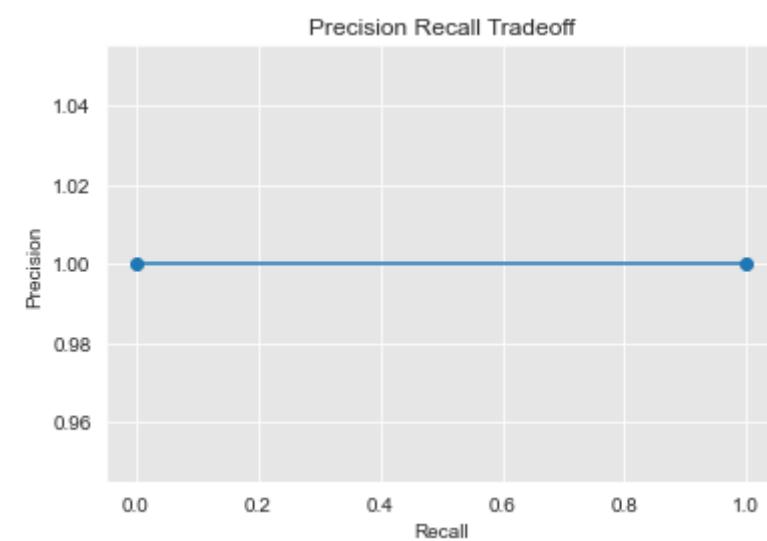
Class:L
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

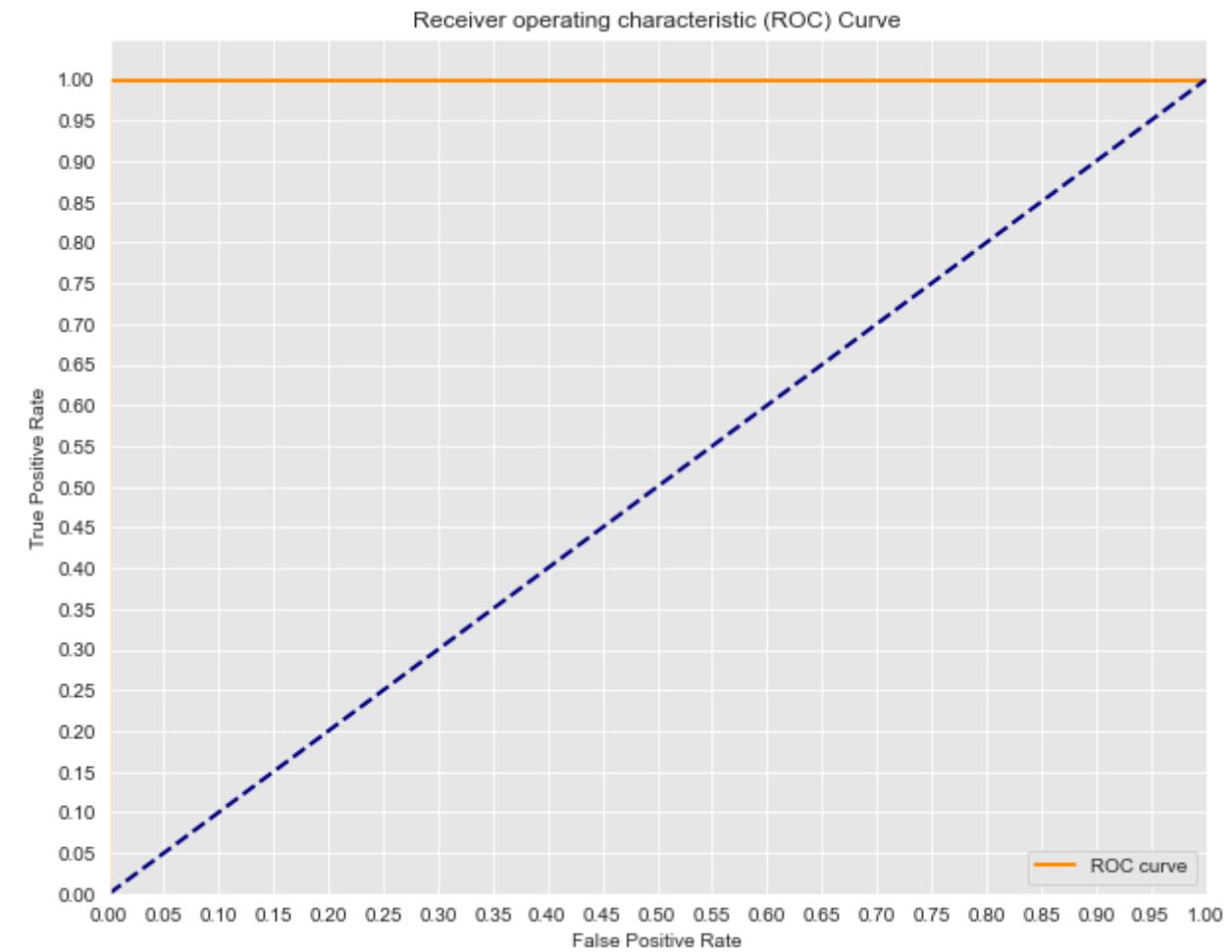




Cross Validated ROC AUC score: 1.0

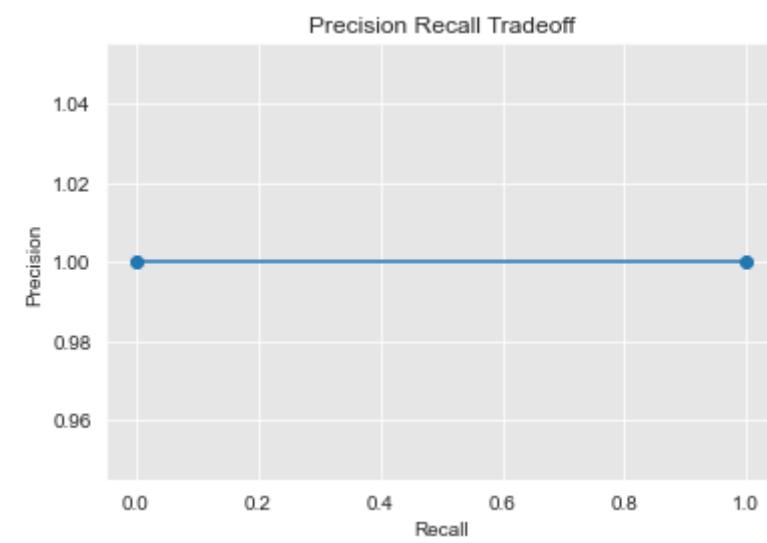
Class:M
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

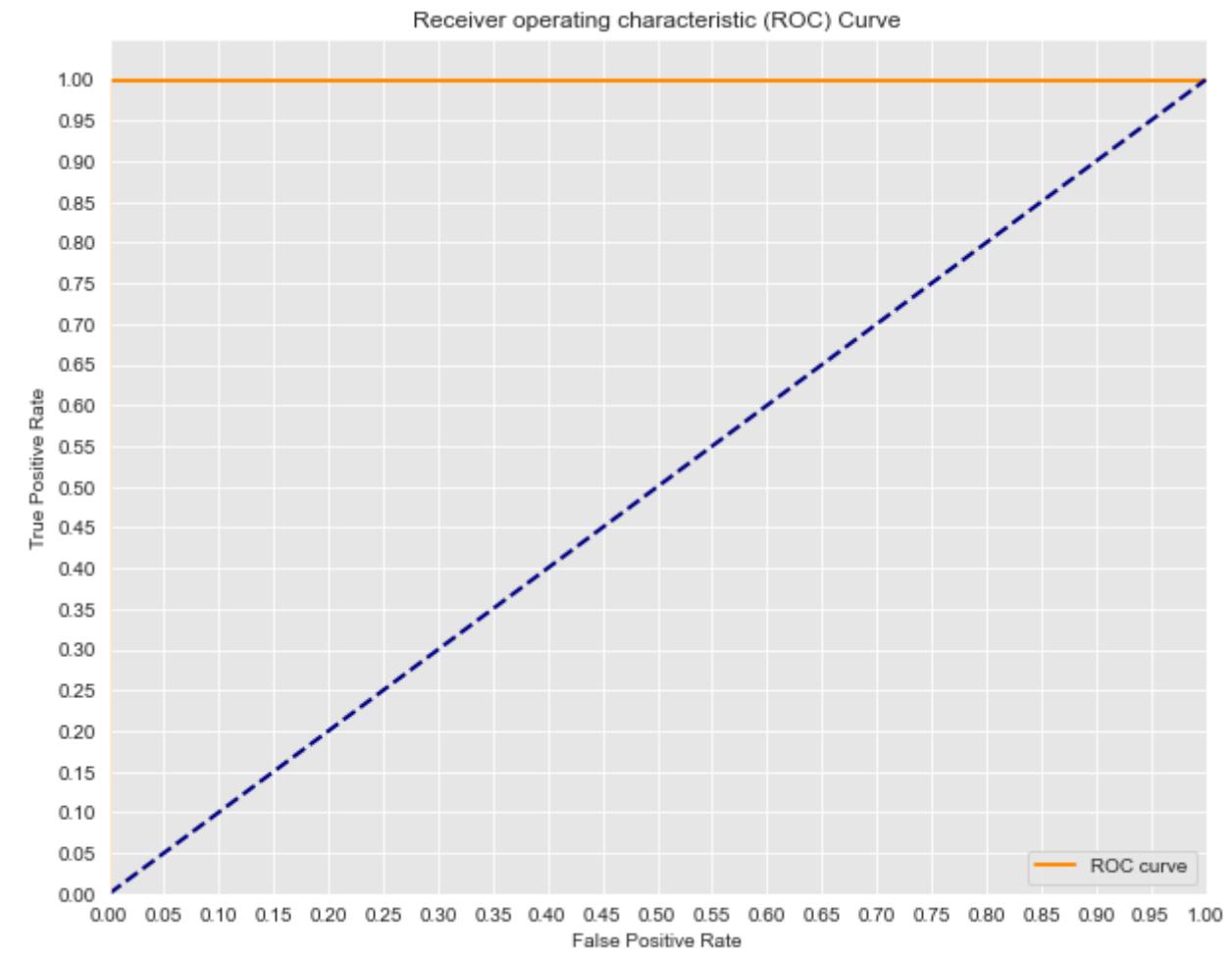




Cross Validated ROC AUC score: 1.0

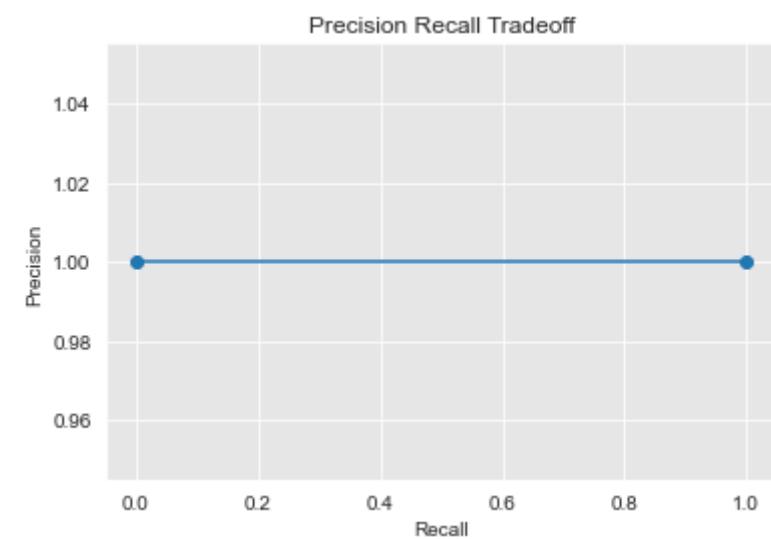
Class:N
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

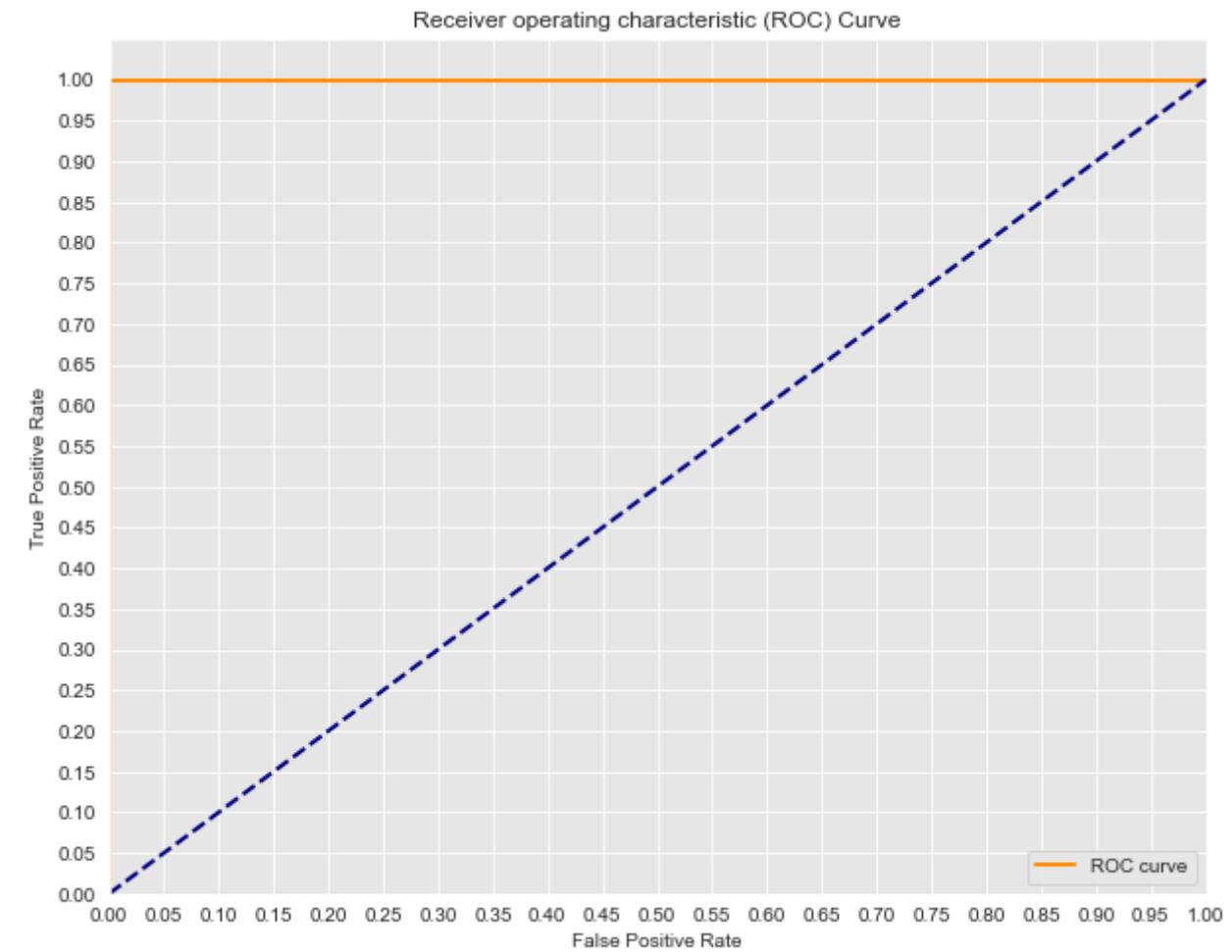




Cross Validated ROC AUC score: 1.0

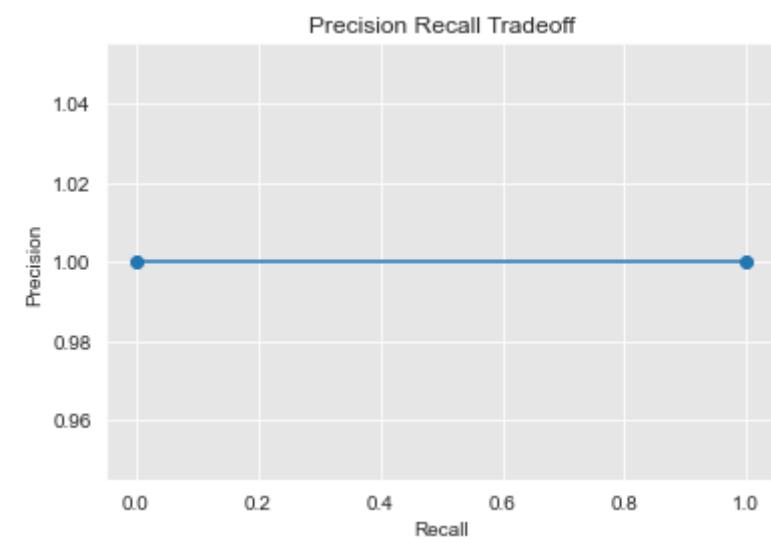
Class:P
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

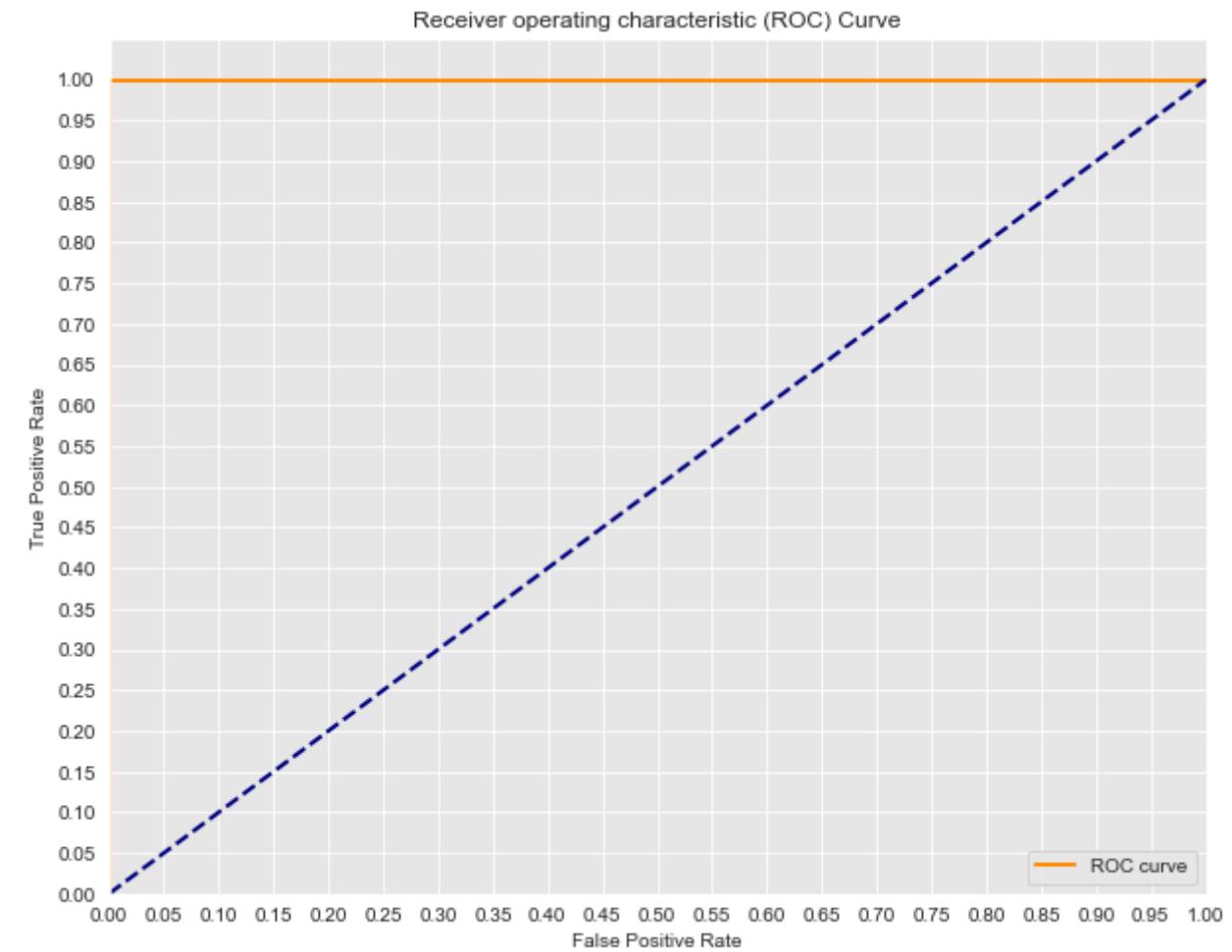




Cross Validated ROC AUC score: 1.0

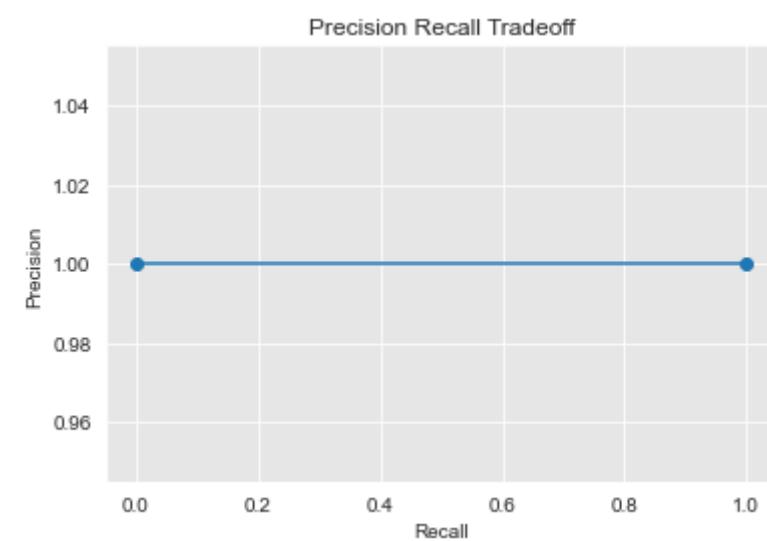
Class:Q
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

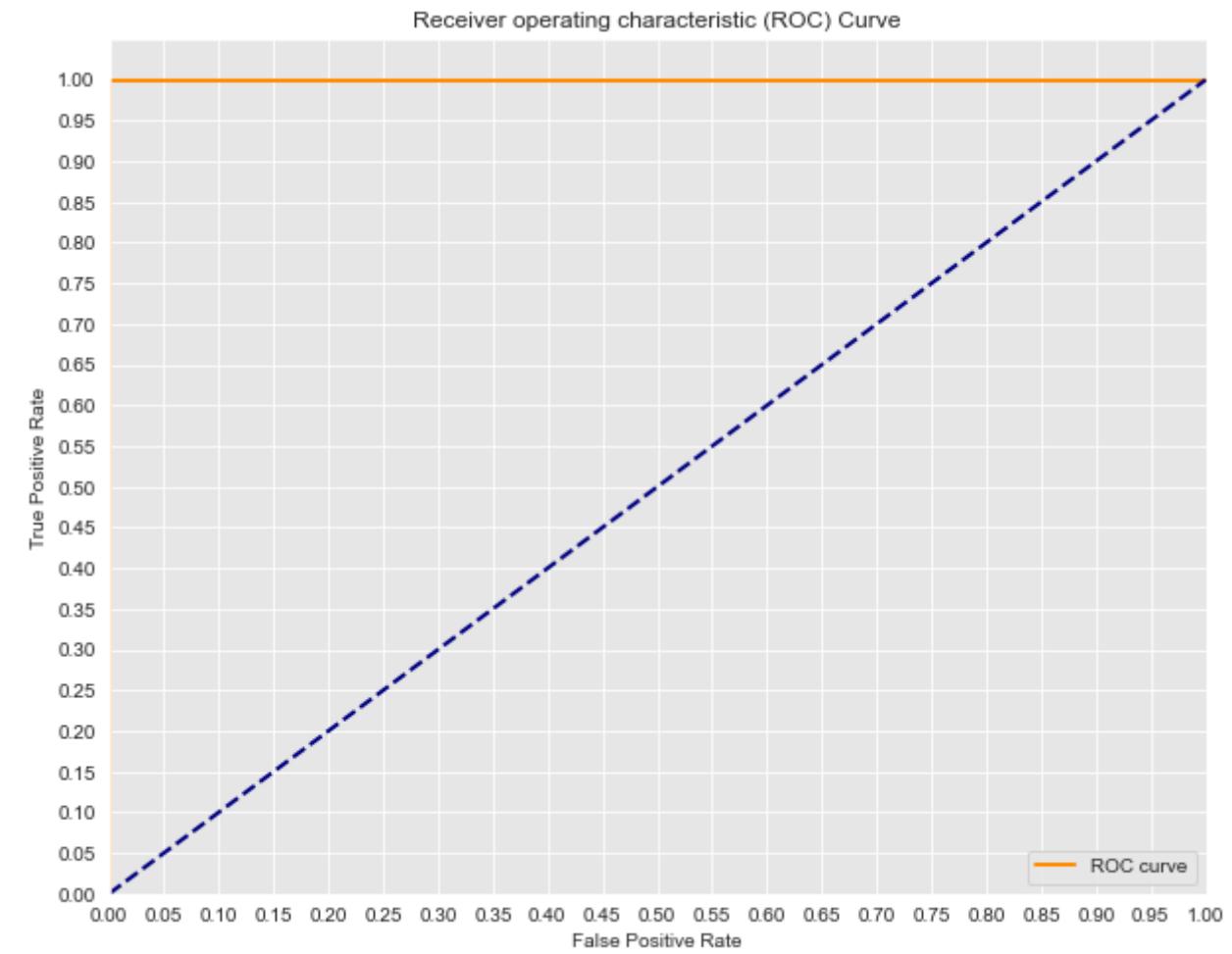




Cross Validated ROC AUC score: 1.0

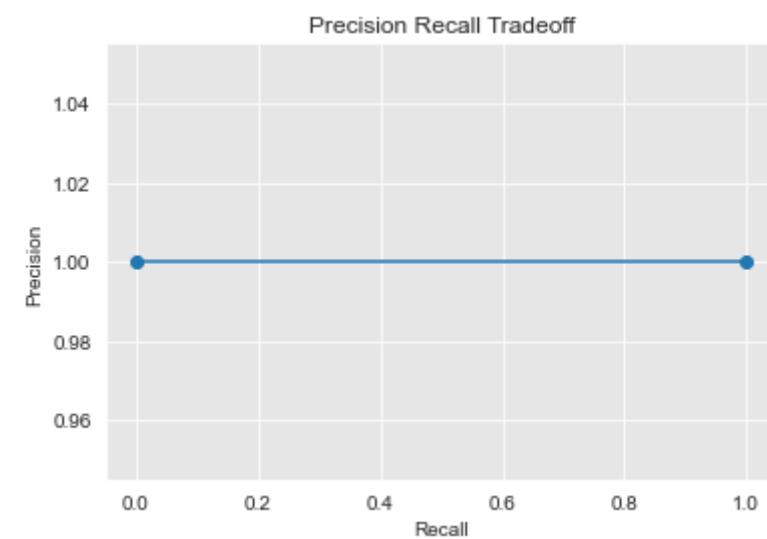
Class:R
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

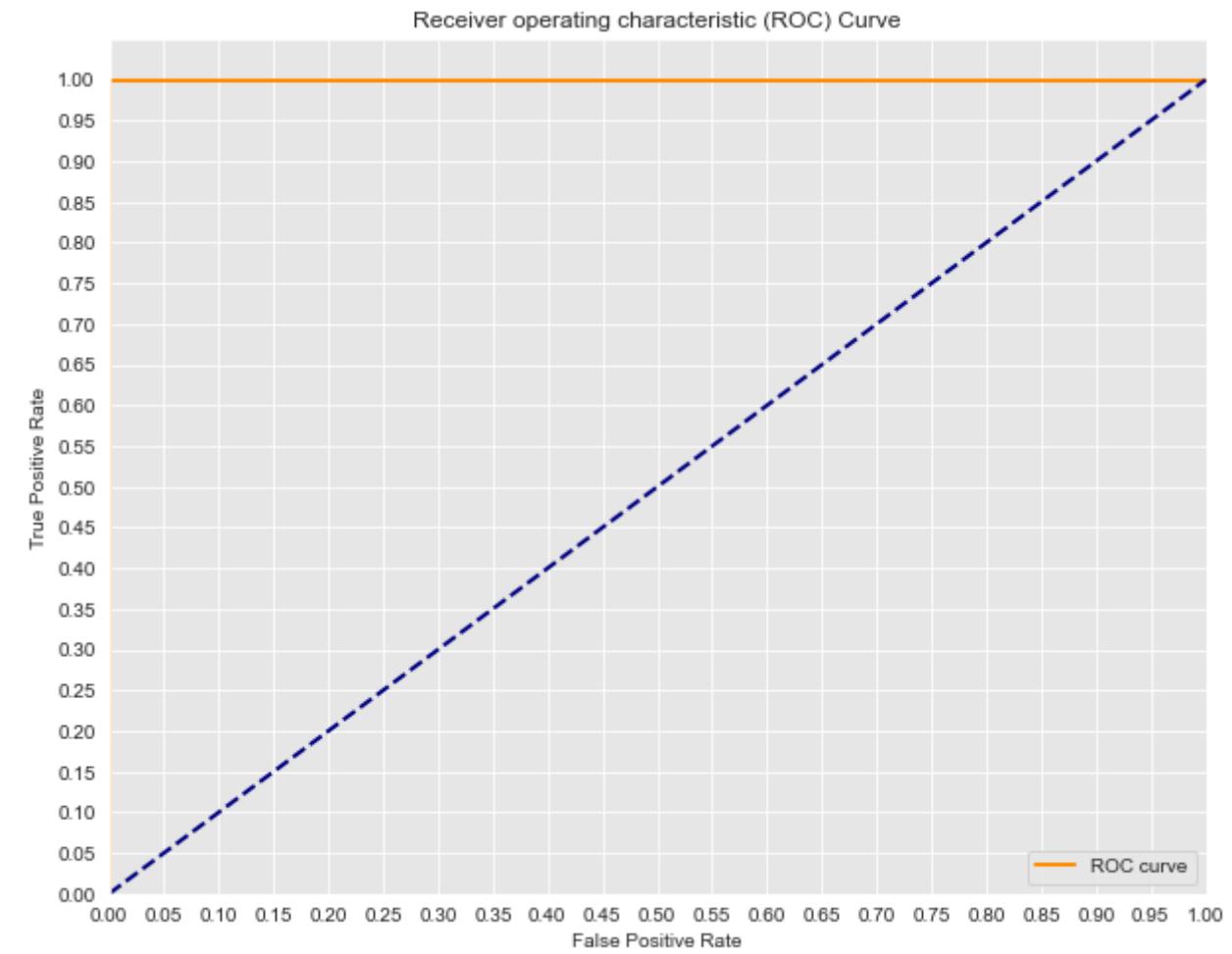




Cross Validated ROC AUC score: 1.0

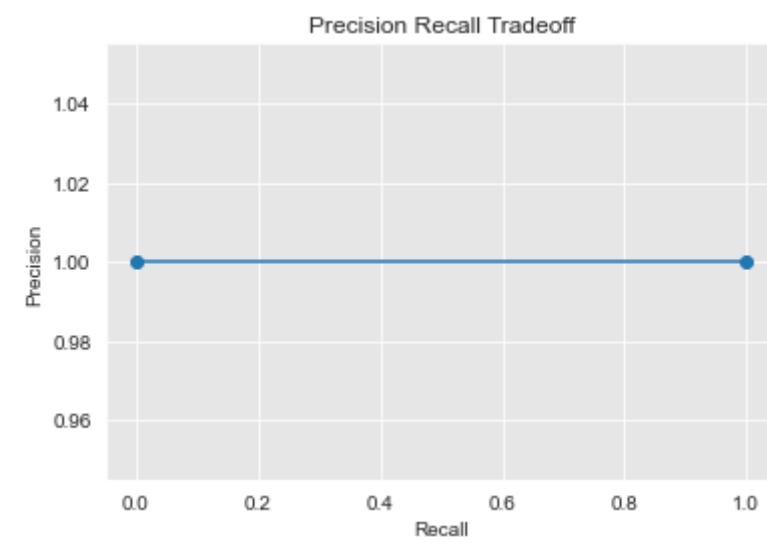
Class:S
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

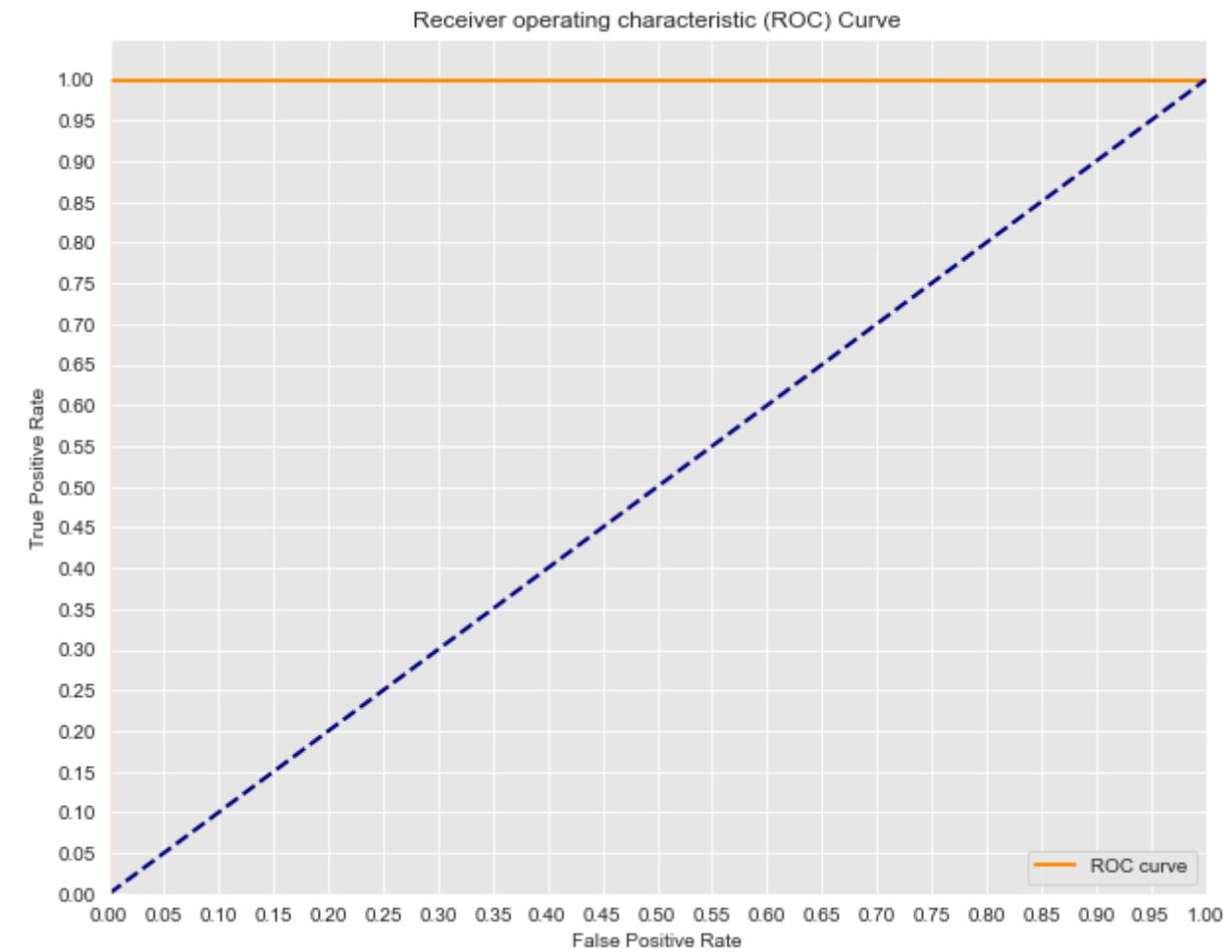




Cross Validated ROC AUC score: 1.0

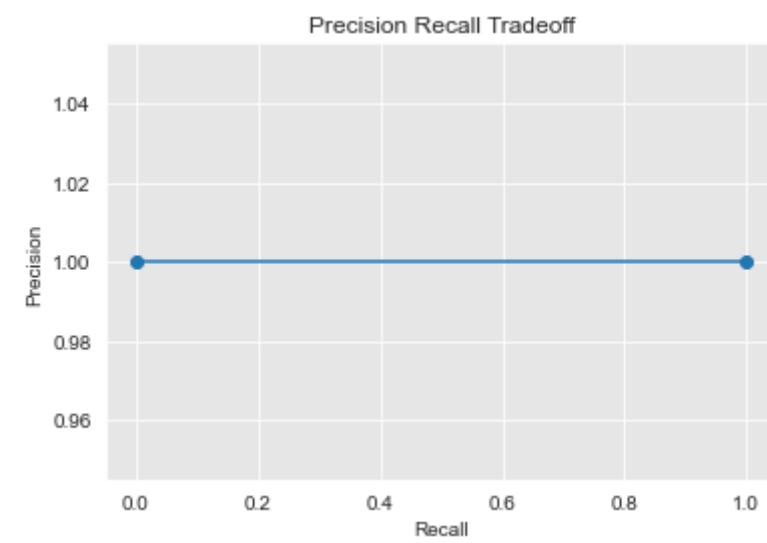
Class:Stop
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

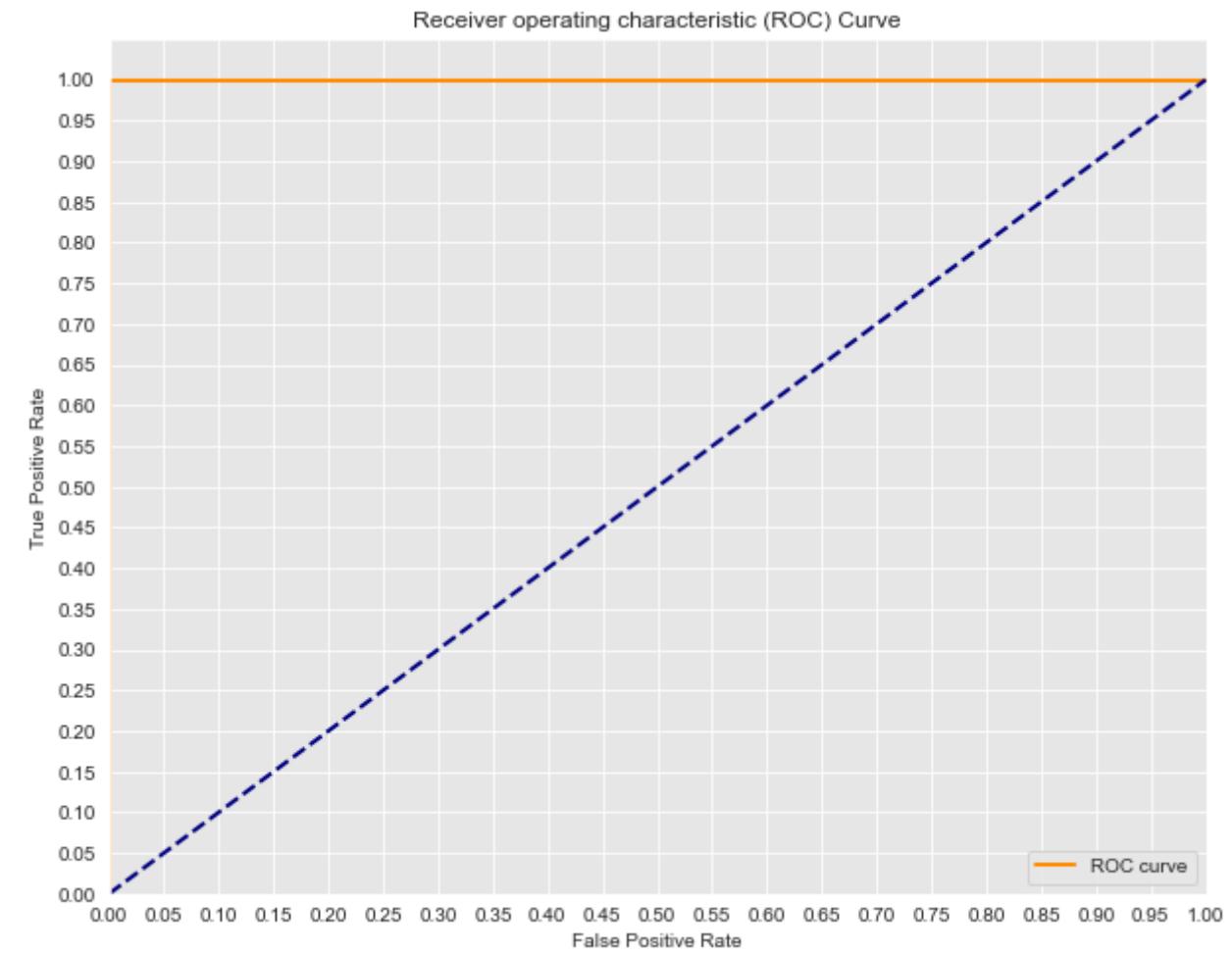




Cross Validated ROC AUC score: 1.0

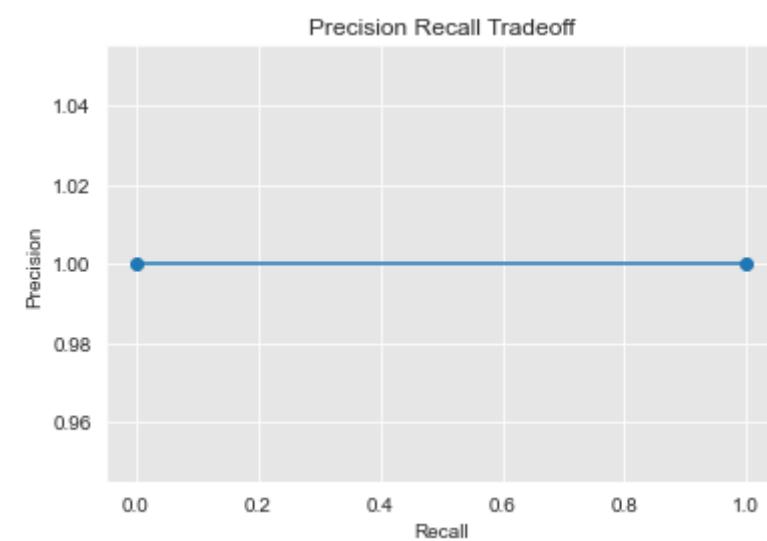
Class:T
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

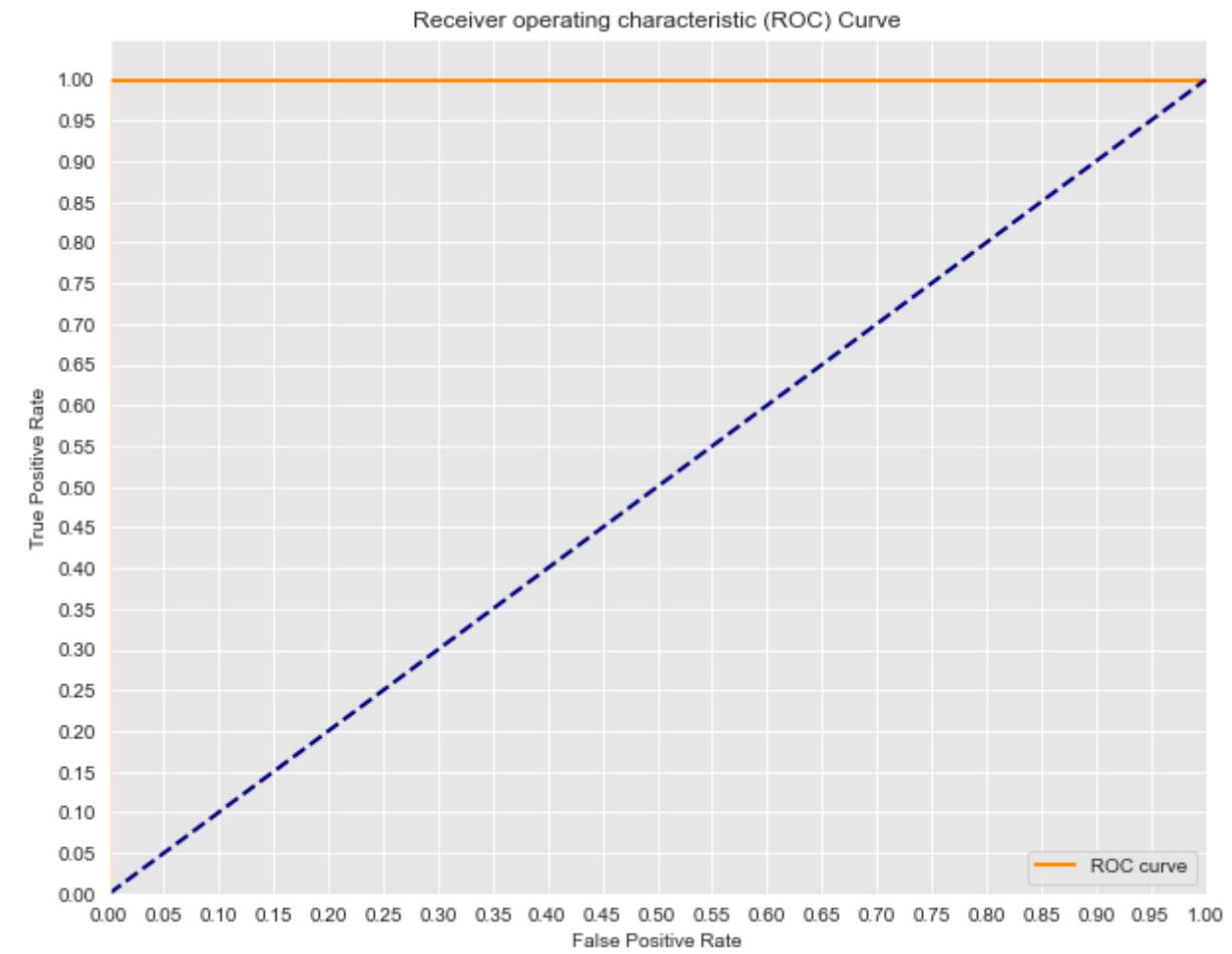




Cross Validated ROC AUC score: 1.0

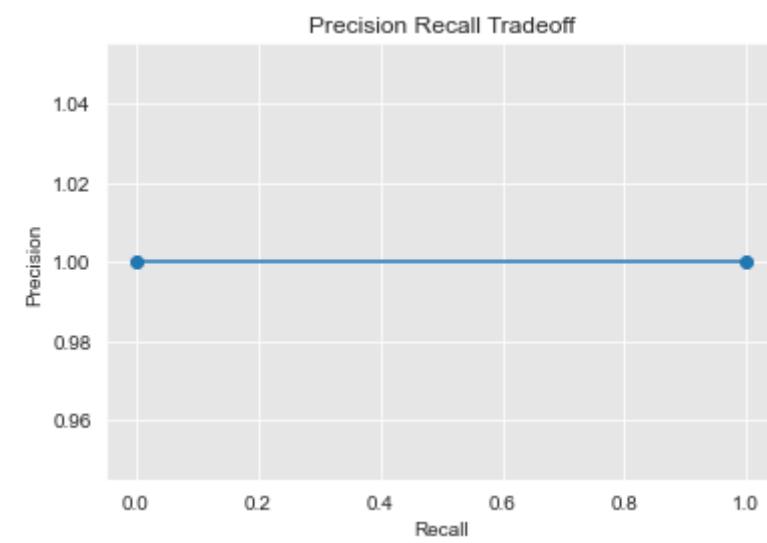
Class:V
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

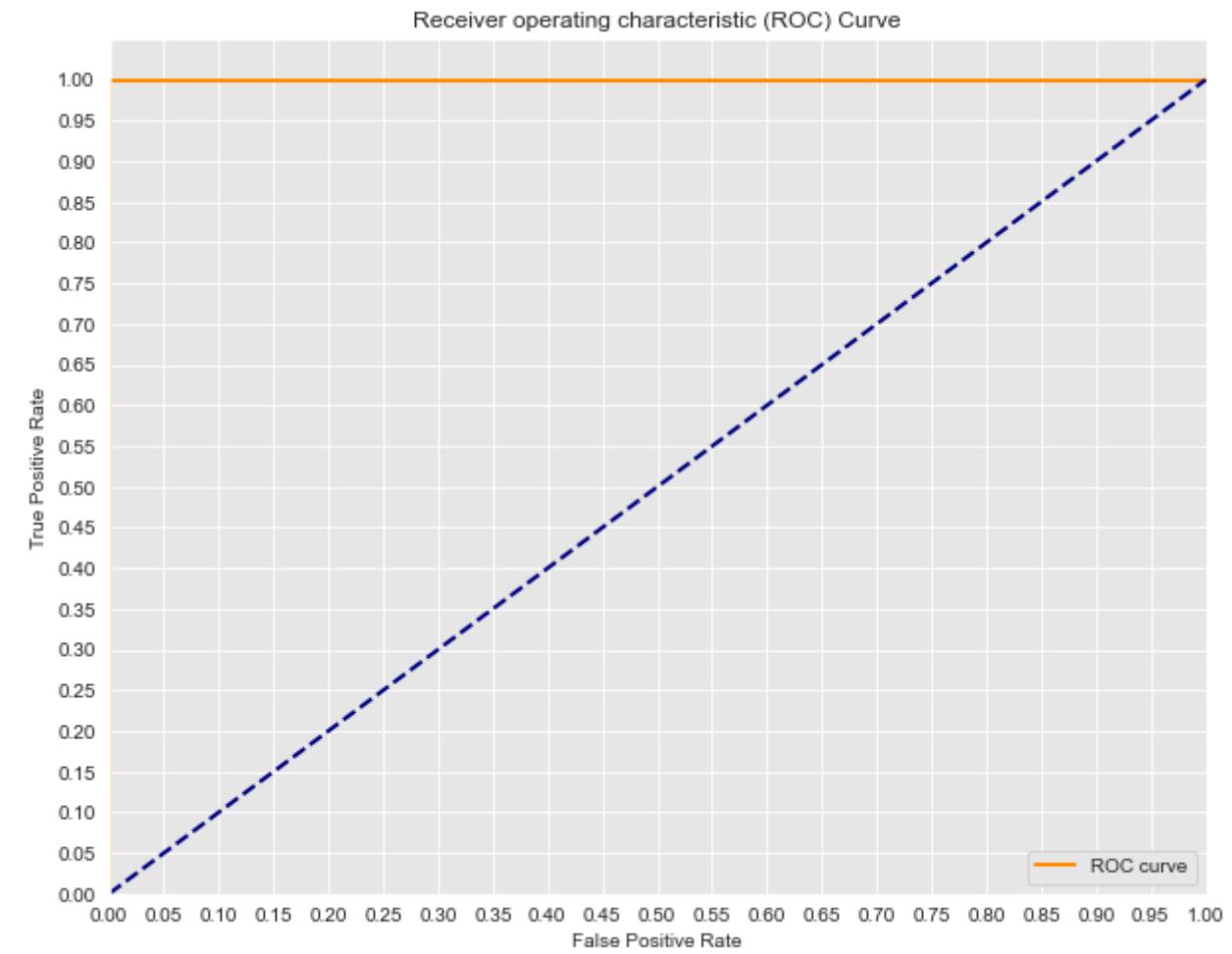




Cross Validated ROC AUC score: 1.0

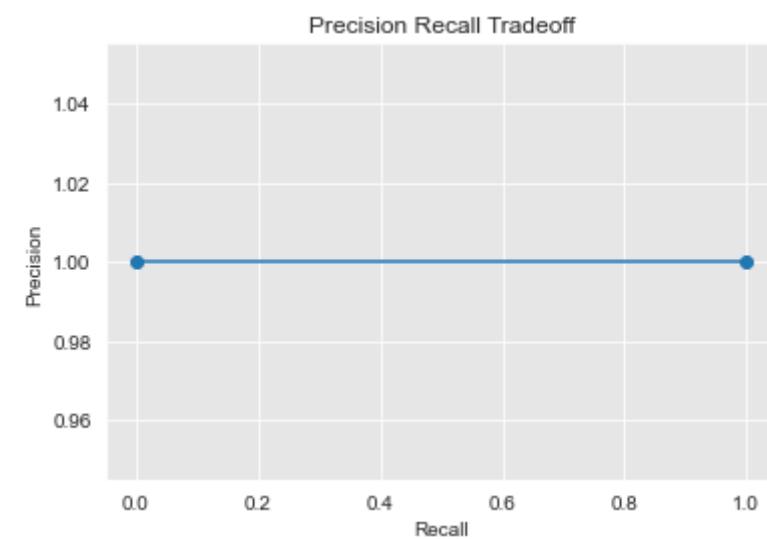
Class:W
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

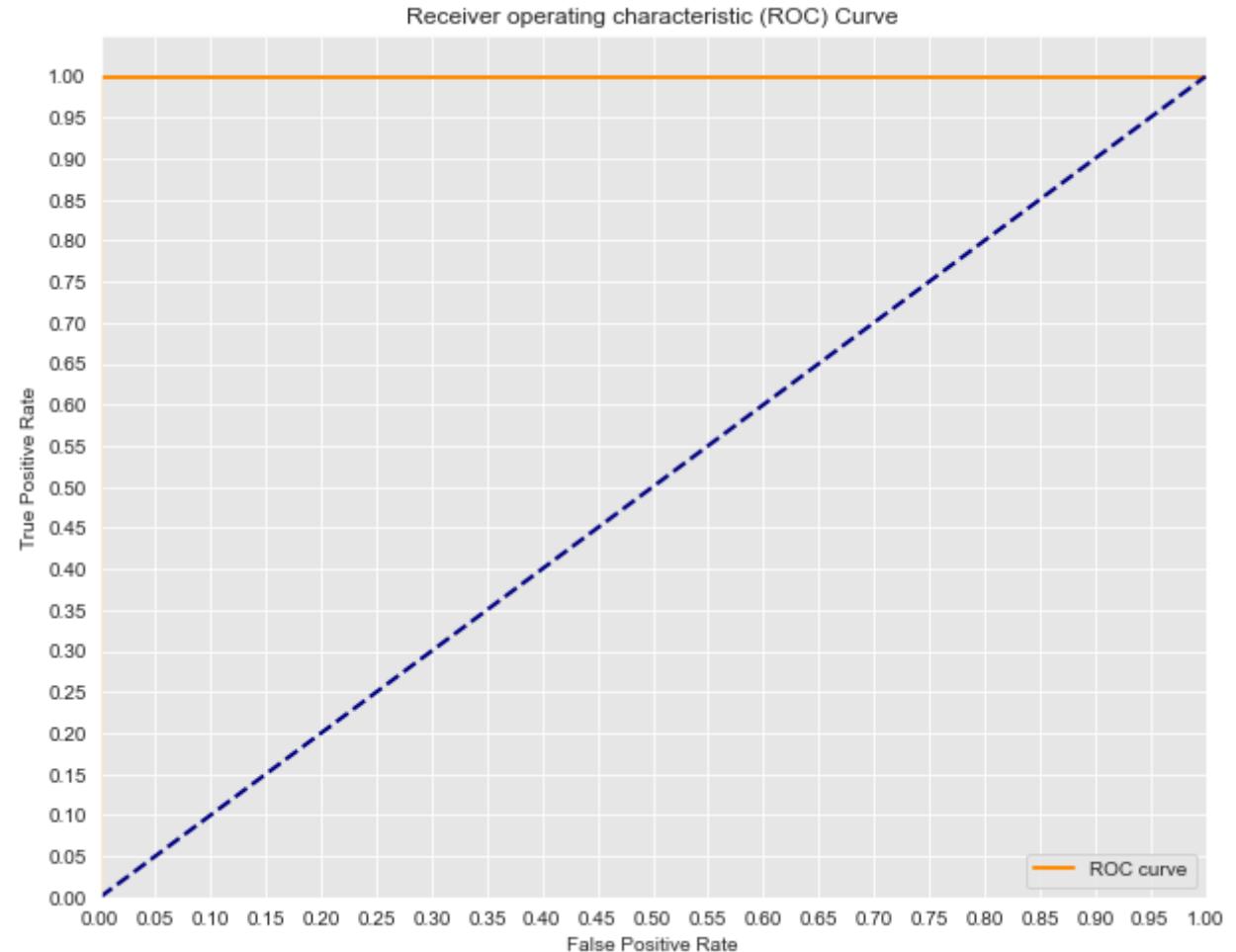




Cross Validated ROC AUC score: 1.0

Class:Y
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0





Cross Validated ROC AUC score: 1.0

All models performed with a cross validated roc auc score of 1. Will test on gradient boost model.

Fit model to test data.

```
In [346]: gradient_boost.fit(x2_test, y2_test)
```

```
Out[346]: GradientBoostingClassifier()
```

Make predictions.

```
In [347]: y2_hat_test_gb = gradient_boost.predict(x2_test)
```

All predictions were classified correctly.

```
In [348]: con_mat(y2_test, y2_hat_test_gb)
```

Confusion Matrix

	0	1	2	3	4	5	6	7	8	9	...	11	12	13	14	15	\
0	76	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
1	0	40	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0
2	0	0	68	0	0	0	0	0	0	0	...	0	0	0	0	0	0
3	0	0	0	118	0	0	0	0	0	0	...	0	0	0	0	0	0
4	0	0	0	0	192	0	0	0	0	0	...	0	0	0	0	0	0
5	0	0	0	0	0	167	0	0	0	0	...	0	0	0	0	0	0
6	0	0	0	0	0	0	31	0	0	0	...	0	0	0	0	0	0
7	0	0	0	0	0	0	0	73	0	0	...	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	135	0	...	0	0	0	0	0	0

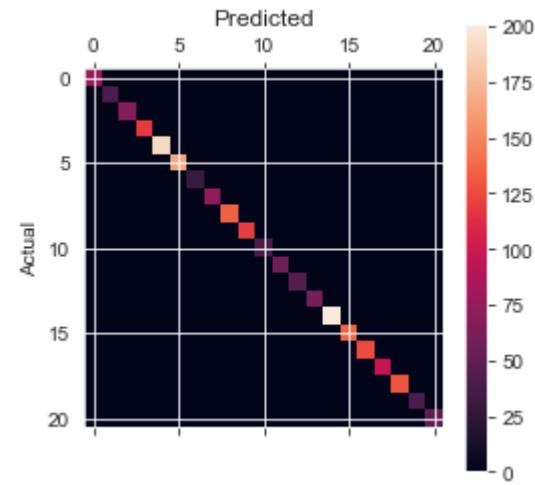
```

9   0   0   0   0   0   0   0   0   0   0   120   ...   0   0   0   0   0
10  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0
11  0   0   0   0   0   0   0   0   0   0   ...   57   0   0   0   0   0
12  0   0   0   0   0   0   0   0   0   0   ...   0   44   0   0   0   0
13  0   0   0   0   0   0   0   0   0   0   ...   0   0   61   0   0   0
14  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   201   0
15  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   136
16  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0   0
17  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0   0
18  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0   0
19  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0   0
20  0   0   0   0   0   0   0   0   0   0   ...   0   0   0   0   0   0

```

	16	17	18	19	20
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0
10	0	0	0	0	0
11	0	0	0	0	0
12	0	0	0	0	0
13	0	0	0	0	0
14	0	0	0	0	0
15	0	0	0	0	0
16	126	0	0	0	0
17	0	97	0	0	0
18	0	0	130	0	0
19	0	0	0	40	0
20	0	0	0	0	49

[21 rows x 21 columns]



Separate actual and predicted values into each class for metrics.

```
In [351]: actual_class0=y2_test==0
actual_class1=y2_test==1
actual_class2=y2_test==2
actual_class3=y2_test==3
actual_class4=y2_test==4
actual_class5=y2_test==5
actual_class6=y2_test==6
```

```

actual_class7=y2_test==7
actual_class8=y2_test==8
actual_class9=y2_test==9
actual_class10=y2_test==10
actual_class11=y2_test==11
actual_class12=y2_test==12
actual_class13=y2_test==13
actual_class14=y2_test==14
actual_class15=y2_test==15
actual_class16=y2_test==16
actual_class17=y2_test==17
actual_class18=y2_test==18
actual_class19=y2_test==19
actual_class20=y2_test==20

predictions_class0=y2_hat_test_gb==0
predictions_class1=y2_hat_test_gb==1
predictions_class2=y2_hat_test_gb==2
predictions_class3=y2_hat_test_gb==3
predictions_class4=y2_hat_test_gb==4
predictions_class5=y2_hat_test_gb==5
predictions_class6=y2_hat_test_gb==6
predictions_class7=y2_hat_test_gb==7
predictions_class8=y2_hat_test_gb==8
predictions_class9=y2_hat_test_gb==9
predictions_class10=y2_hat_test_gb==10
predictions_class11=y2_hat_test_gb==11
predictions_class12=y2_hat_test_gb==12
predictions_class13=y2_hat_test_gb==13
predictions_class14=y2_hat_test_gb==14
predictions_class15=y2_hat_test_gb==15
predictions_class16=y2_hat_test_gb==16
predictions_class17=y2_hat_test_gb==17
predictions_class18=y2_hat_test_gb==18
predictions_class19=y2_hat_test_gb==19
predictions_class20=y2_hat_test_gb==20

a=[actual_class0,actual_class1,actual_class2,actual_class3,actual_class4,actual_class5,actual_class6,actual_class7,
   actual_class8,actual_class9,actual_class10,actual_class11,actual_class12,actual_class13,actual_class14,actual_class15
   ,actual_class16,actual_class17,actual_class18,actual_class19,actual_class20]
p=[predictions_class0,predictions_class1,predictions_class2,predictions_class3,predictions_class4,predictions_class5,predictions_class6
   ,predictions_class7,predictions_class8,predictions_class9,predictions_class10,predictions_class11,predictions_class12
   ,predictions_class13,predictions_class14,predictions_class15,predictions_class16,predictions_class17,predictions_class18
   ,predictions_class19,predictions_class20]

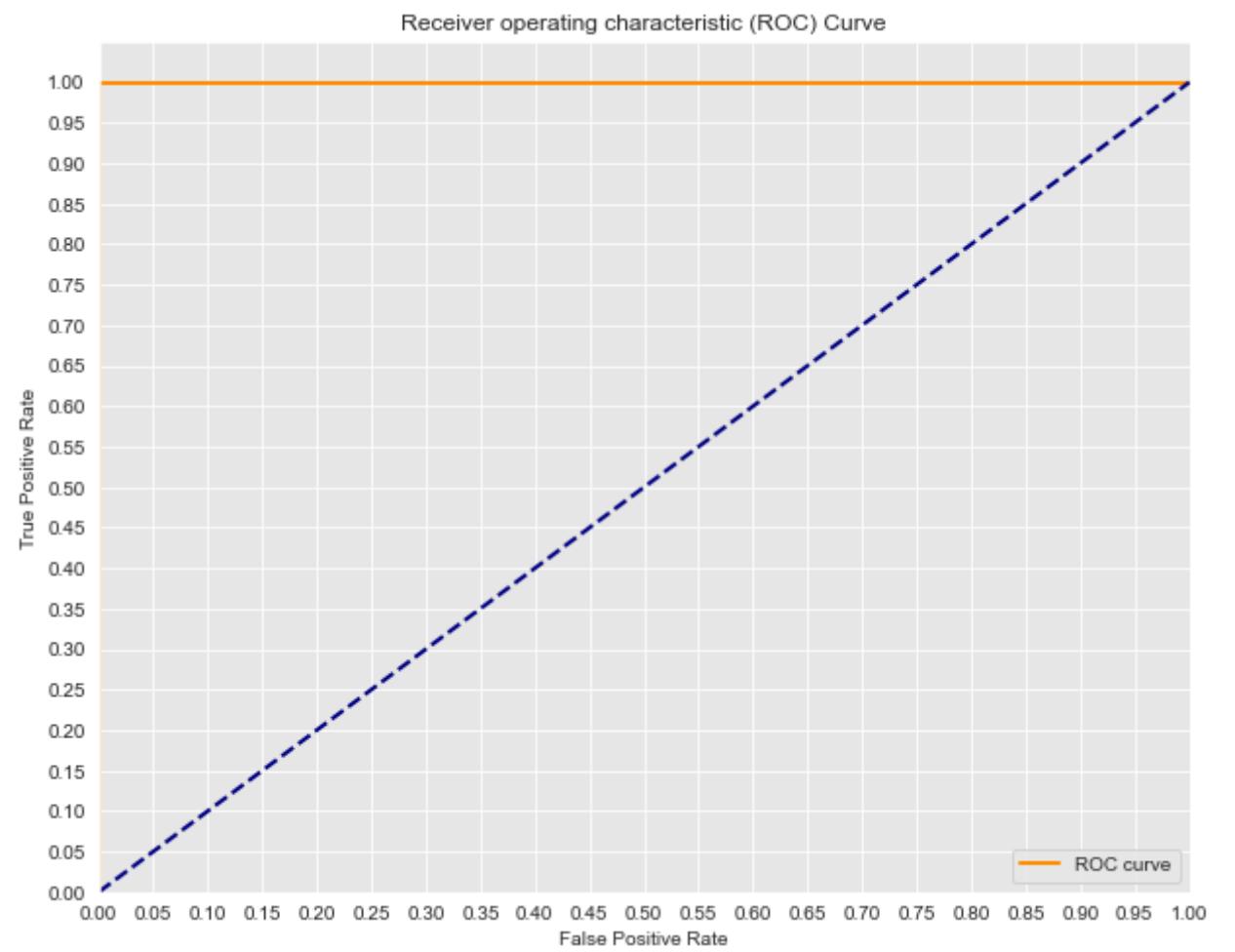
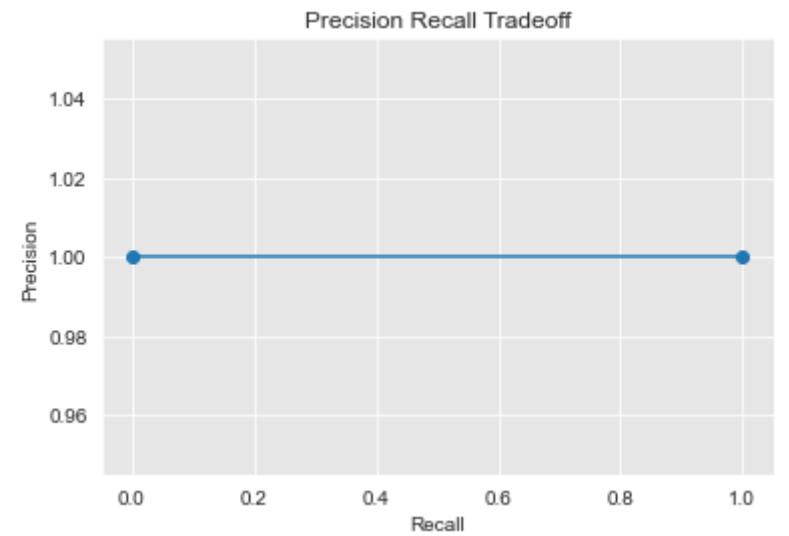
```

In [352...]: `multiclass(gradient_boost,x2_test,a,p,translation_df2['Amino_Acids_copy'])`

```

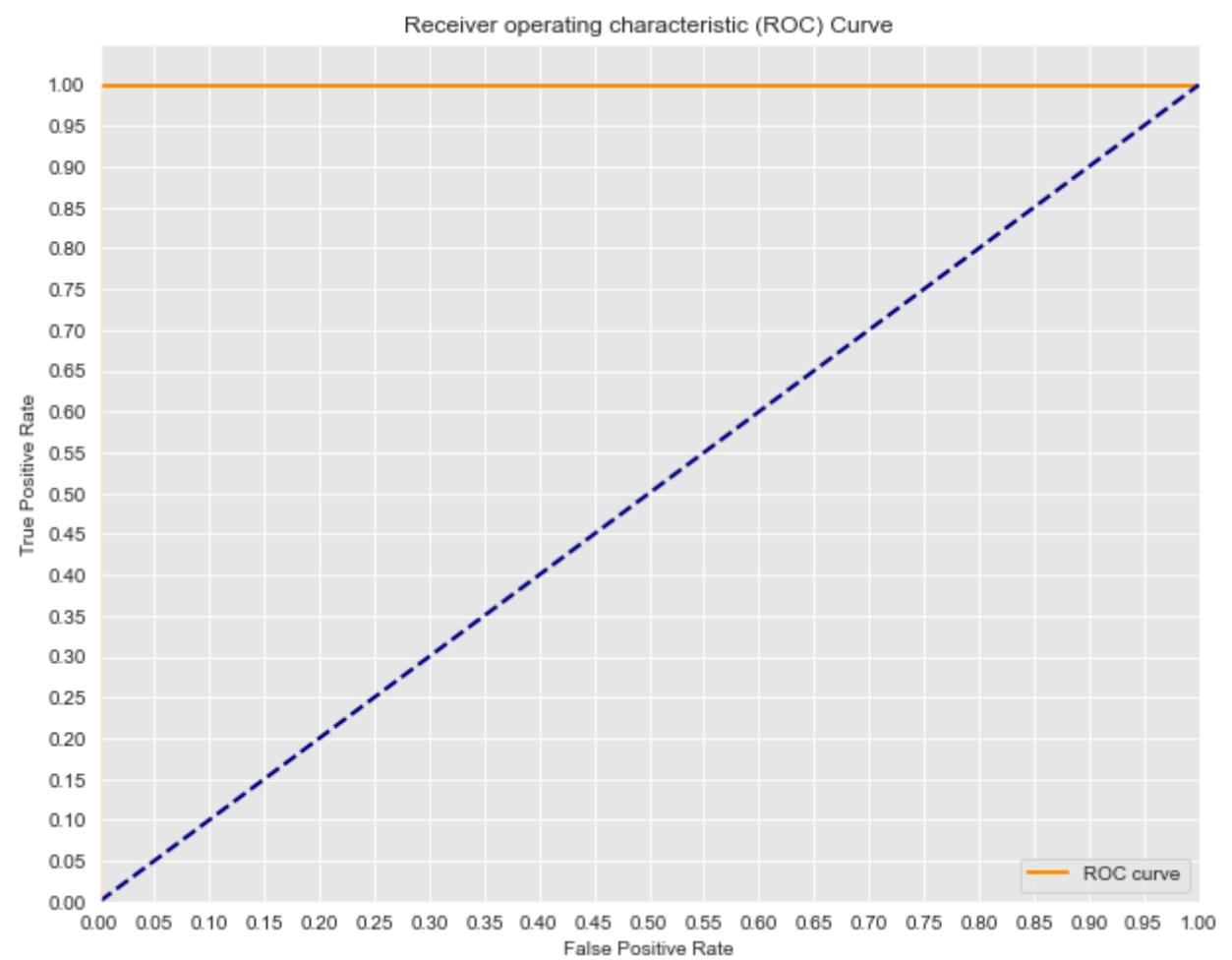
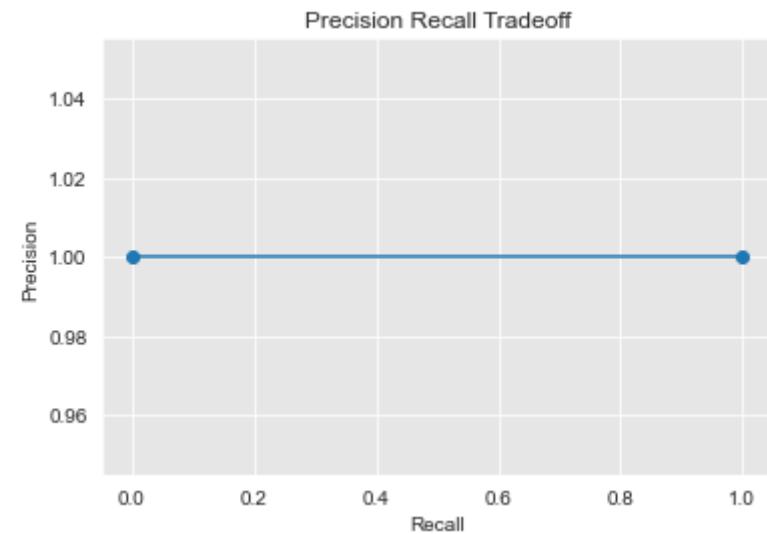
Class:A
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0

```



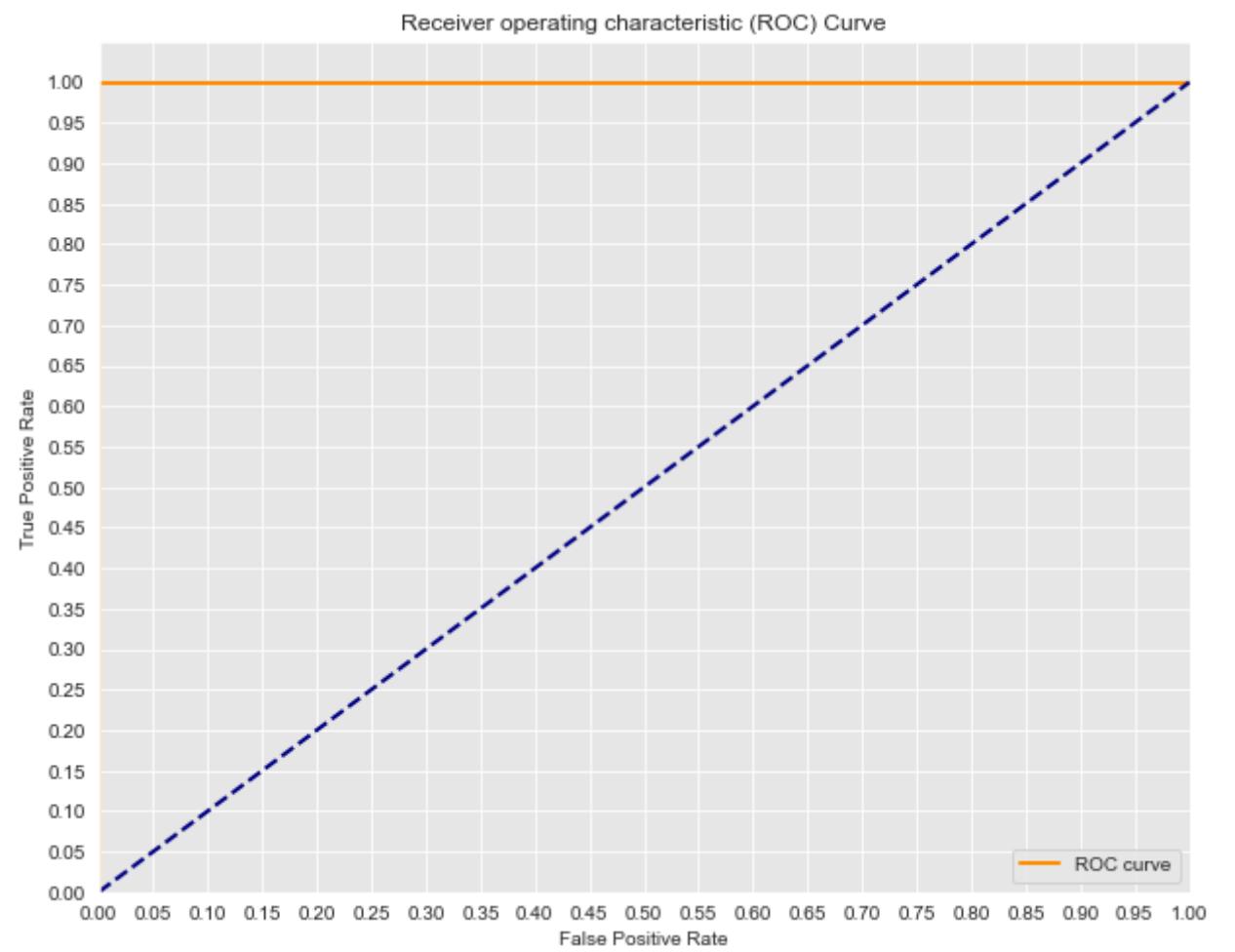
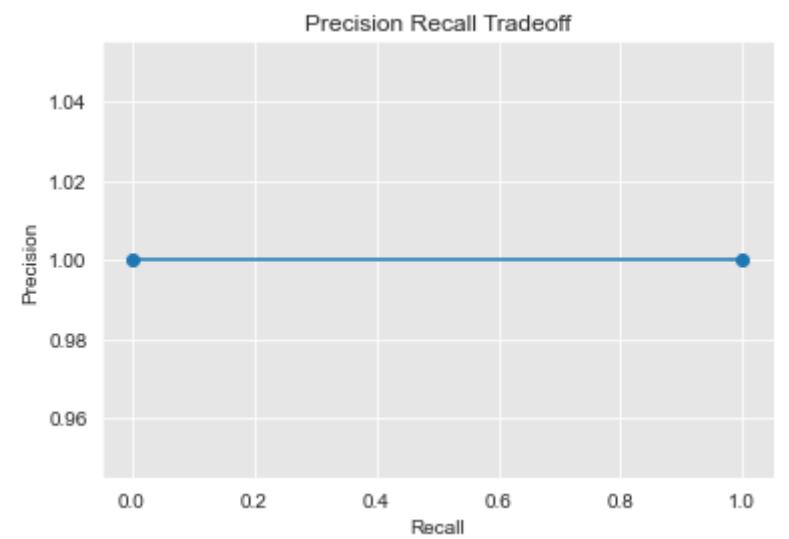
Cross Validated ROC AUC score: 1.0

Class:C
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



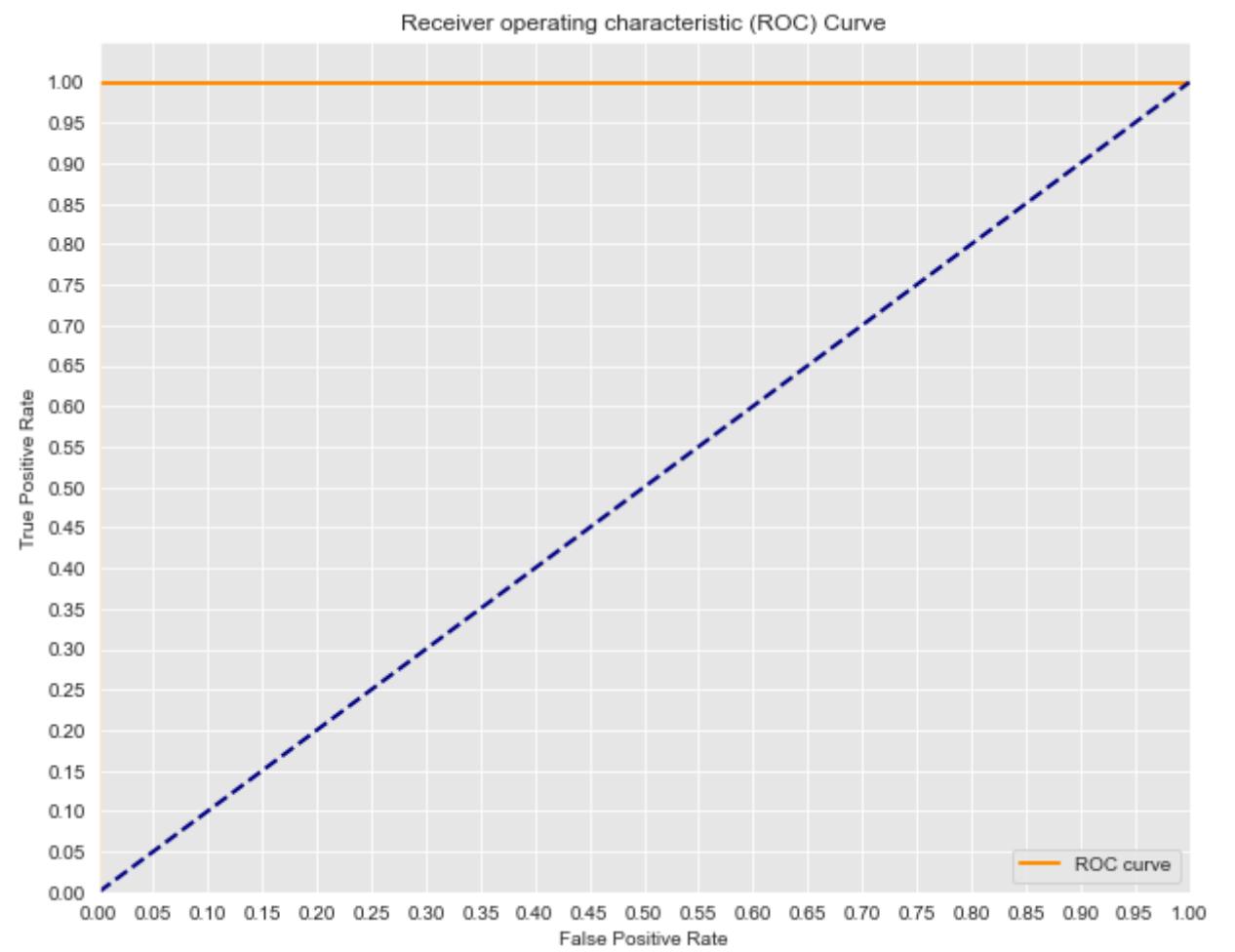
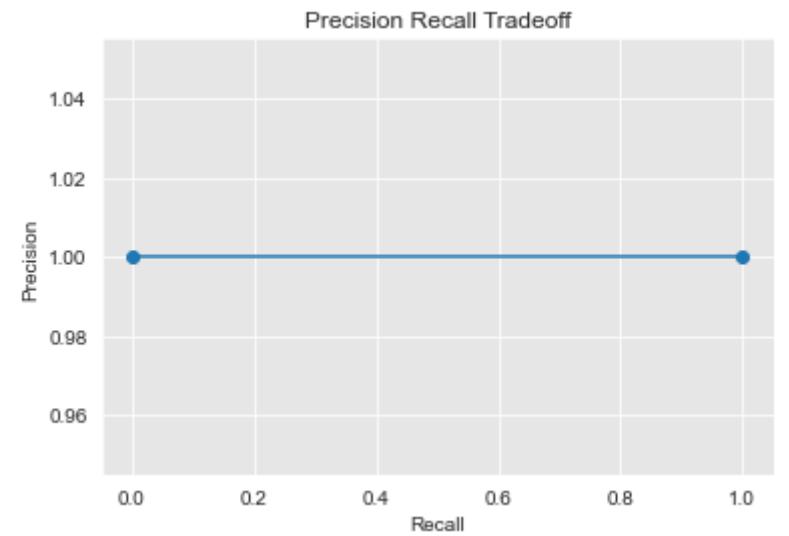
Cross Validated ROC AUC score: 1.0

Class:D
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



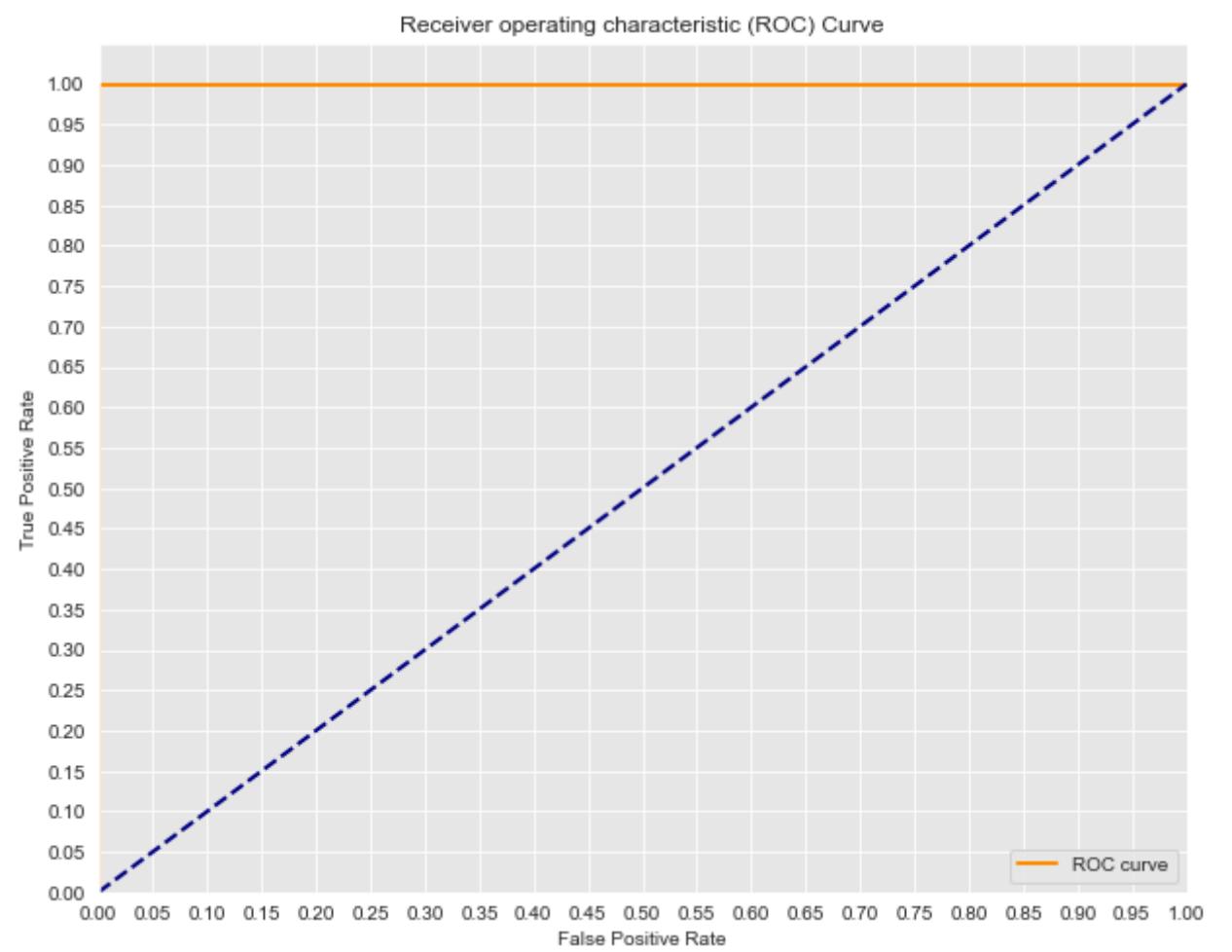
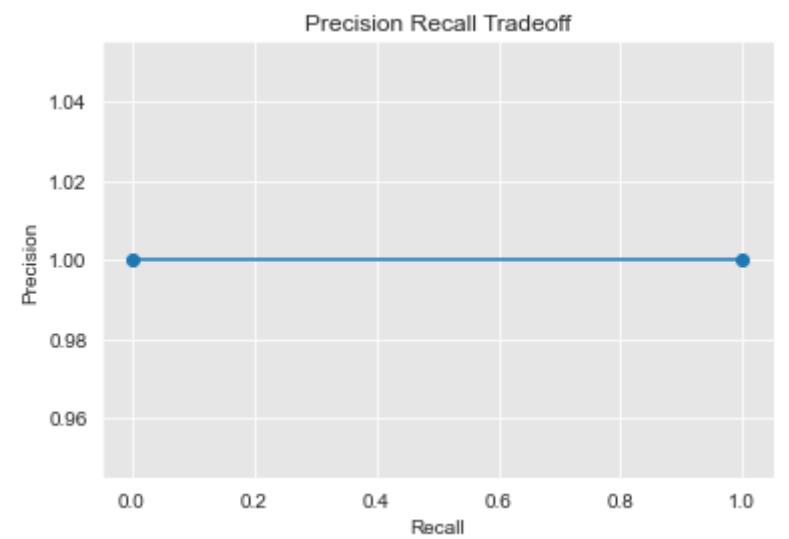
Cross Validated ROC AUC score: 1.0

Class:E
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



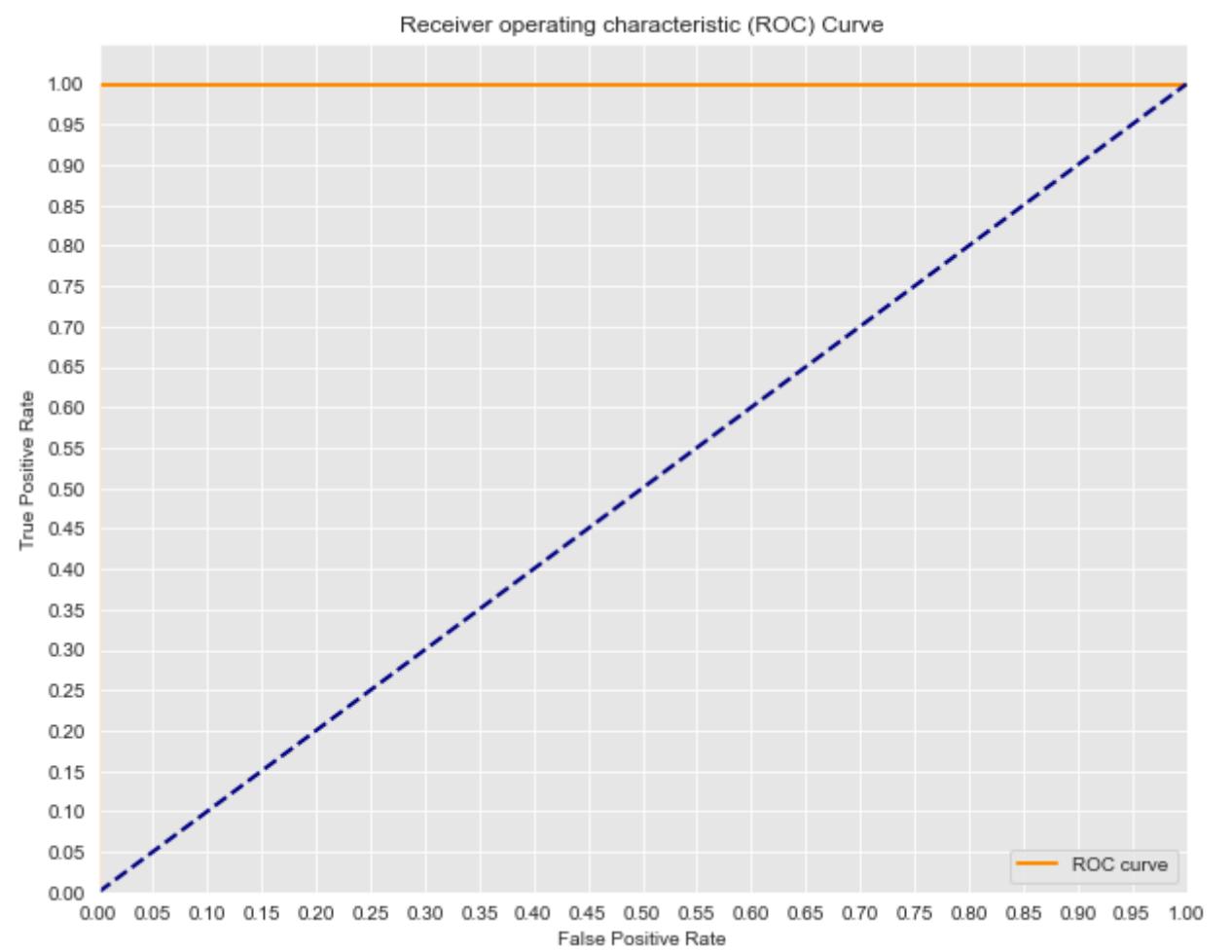
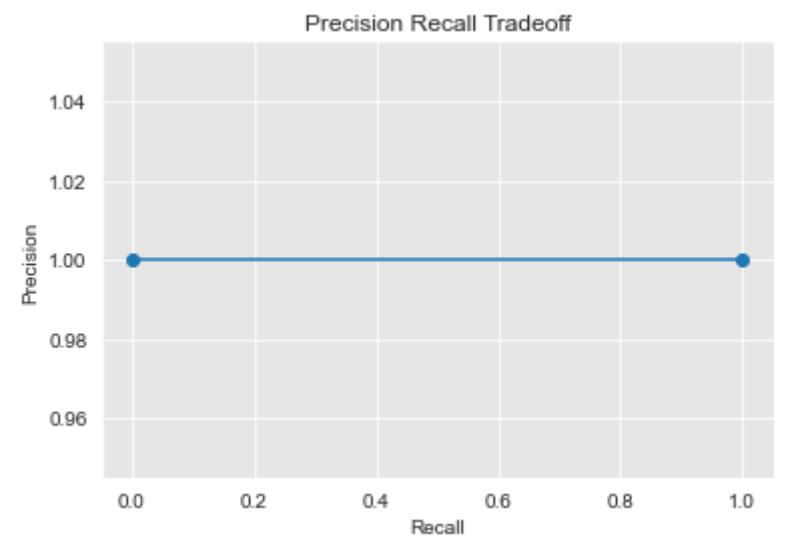
Cross Validated ROC AUC score: 1.0

Class:F
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



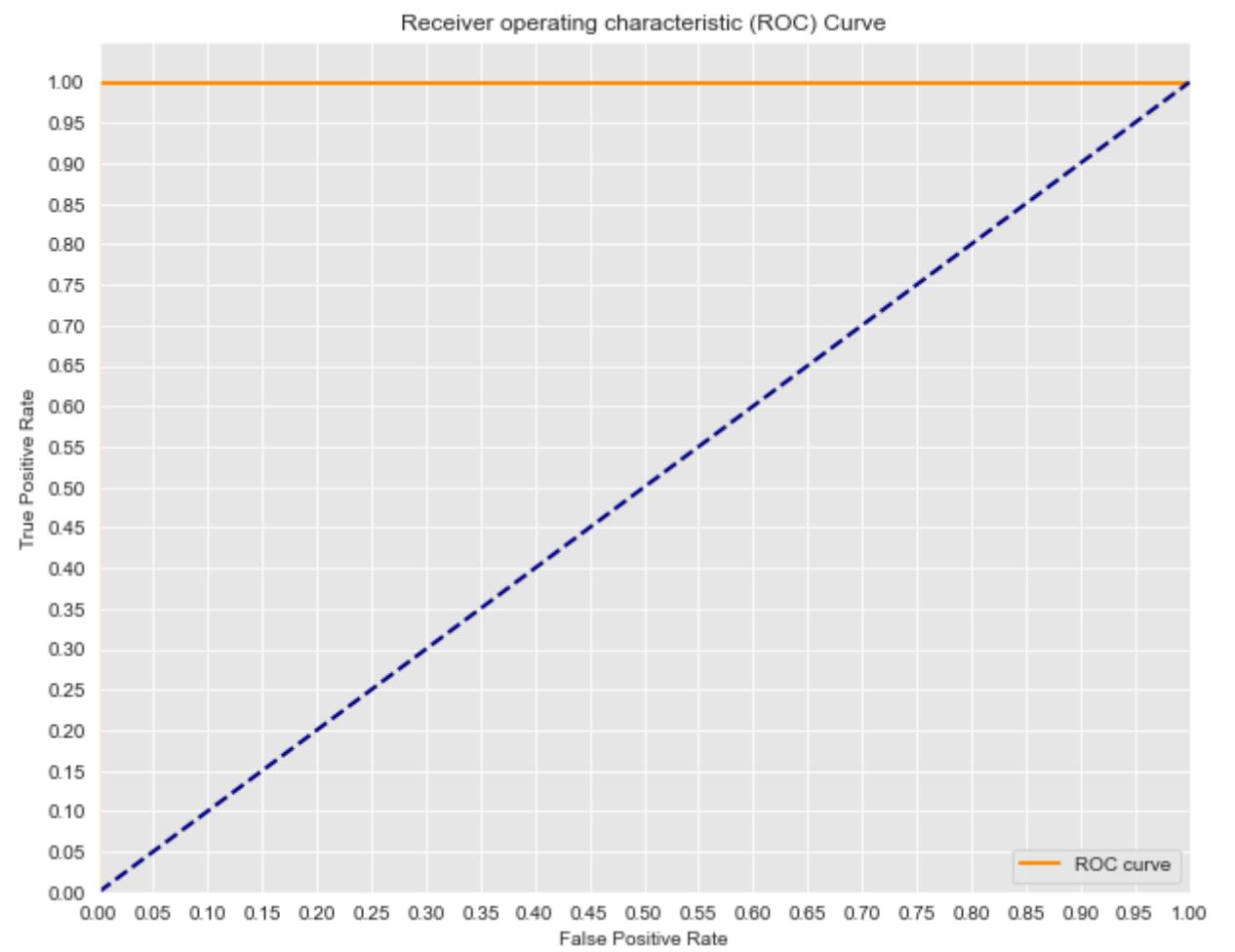
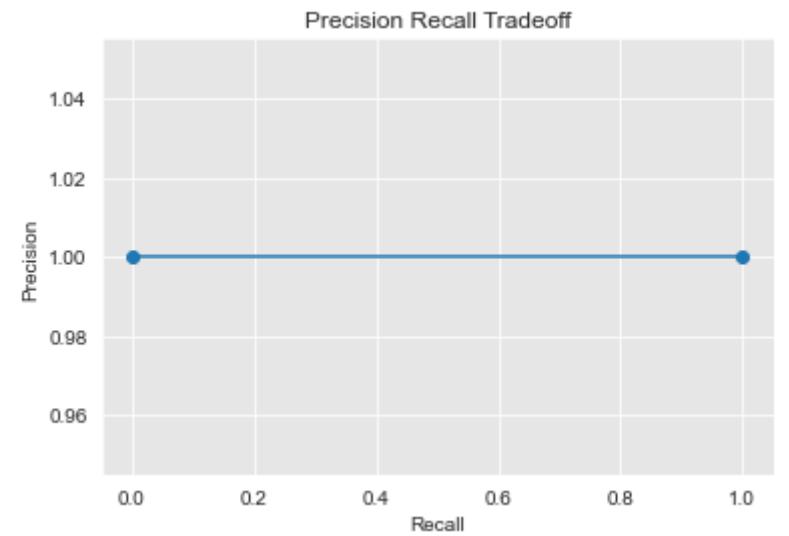
Cross Validated ROC AUC score: 1.0

Class:G
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



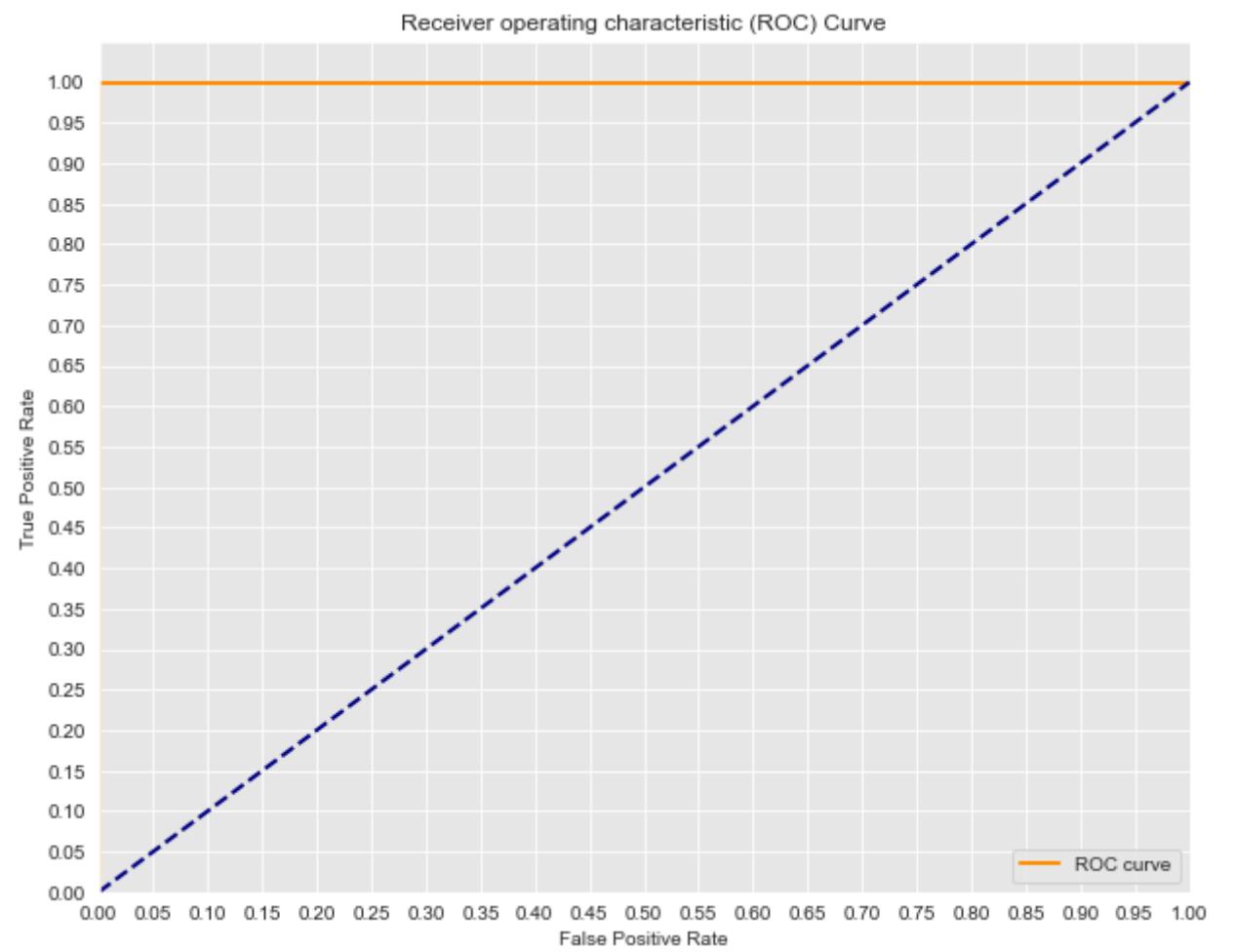
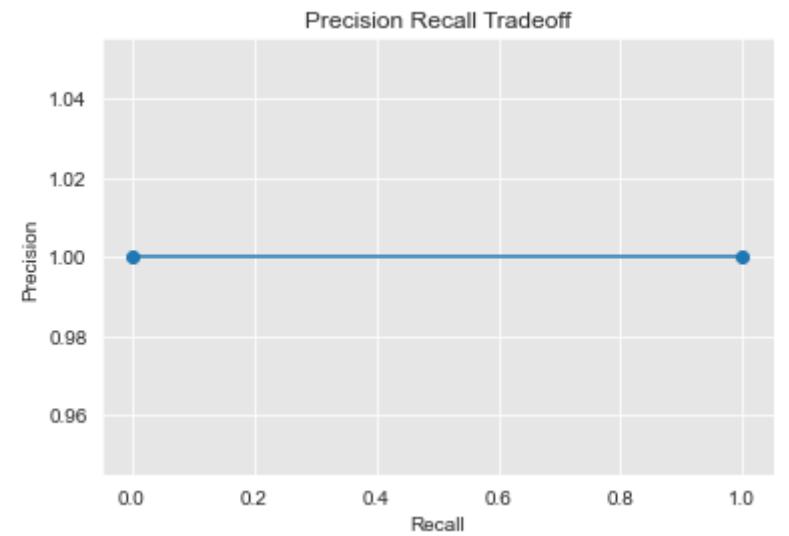
Cross Validated ROC AUC score: 1.0

Class:H
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



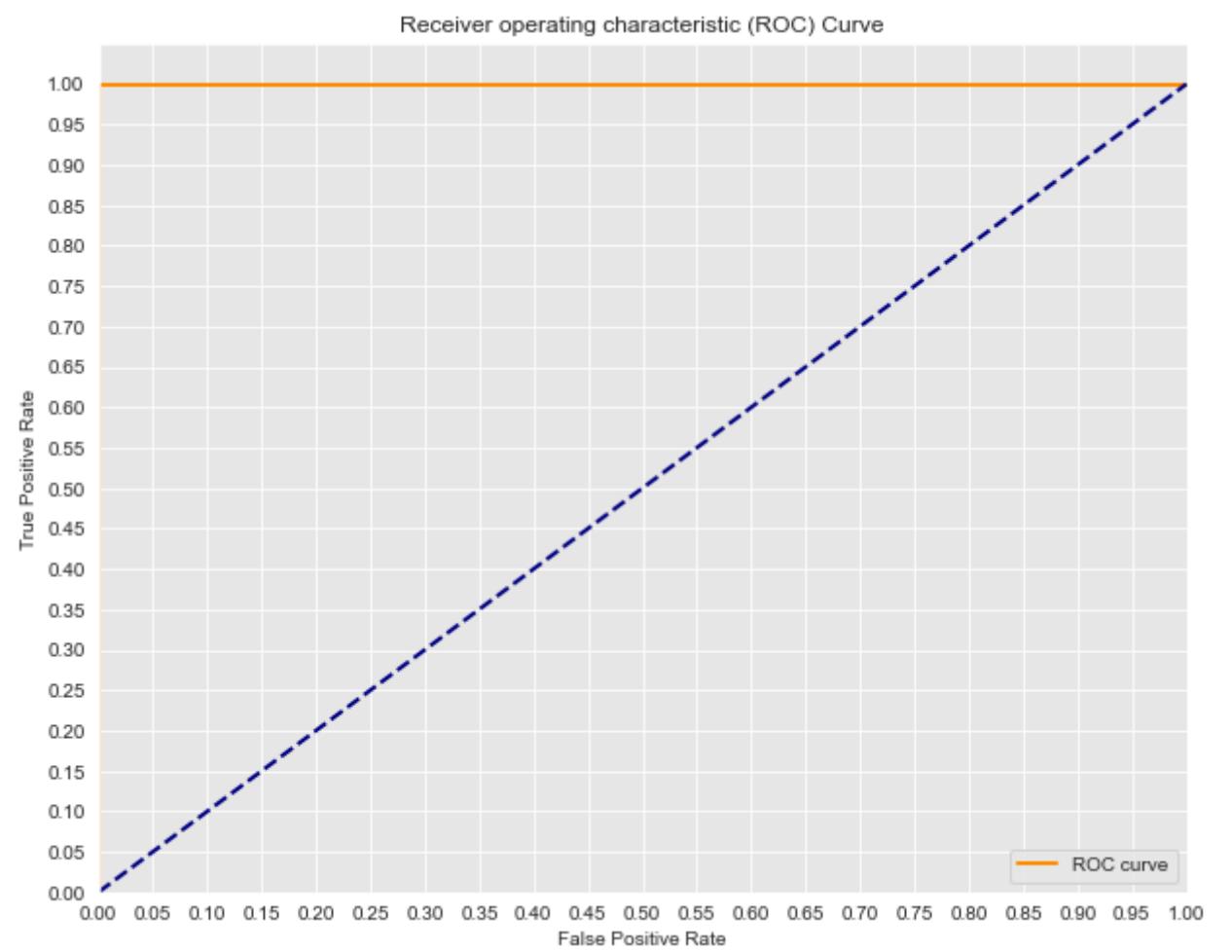
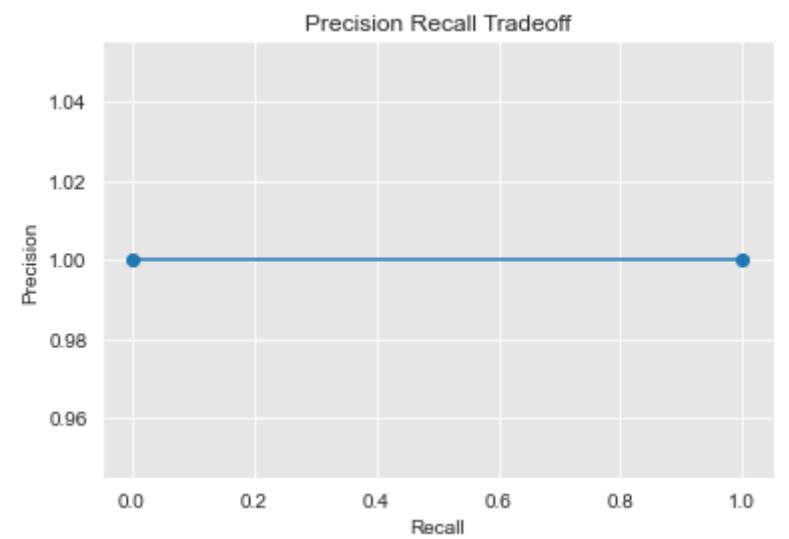
Cross Validated ROC AUC score: 1.0

Class:I
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



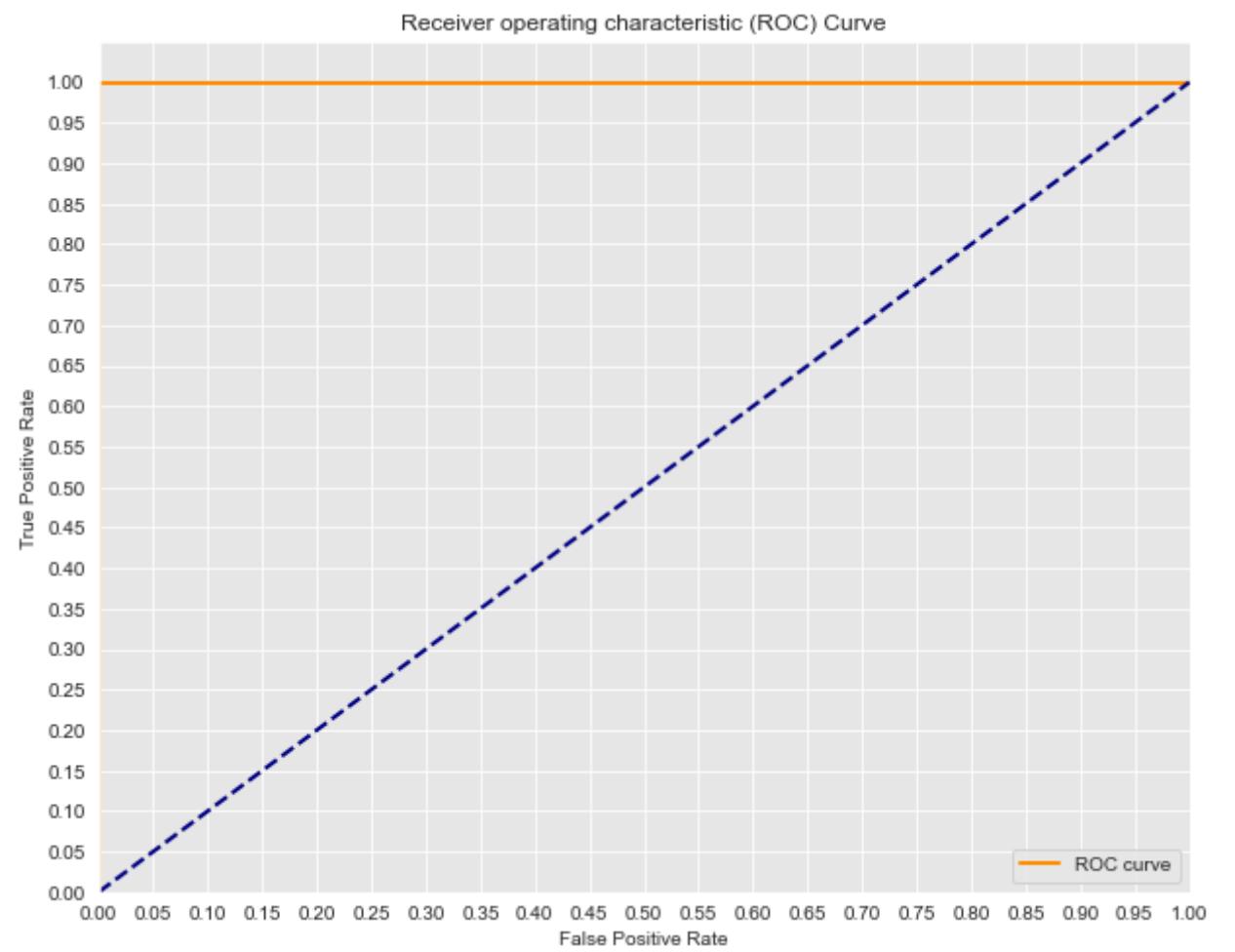
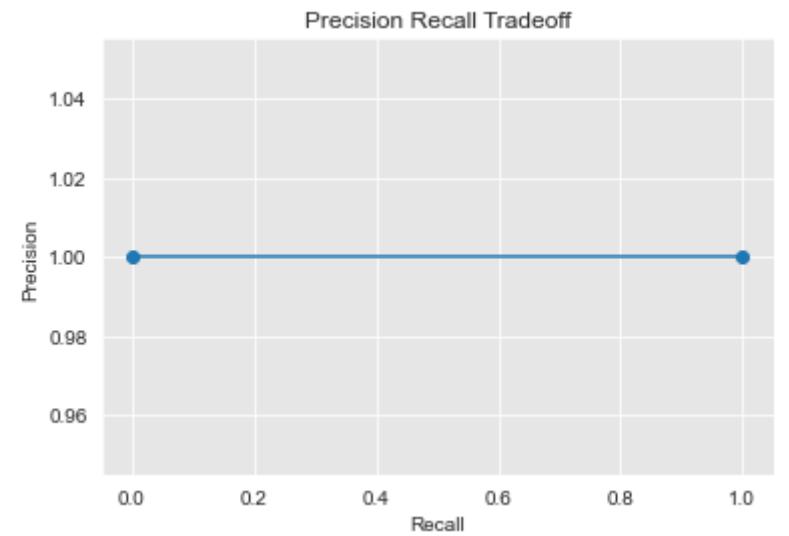
Cross Validated ROC AUC score: 1.0

Class:K
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



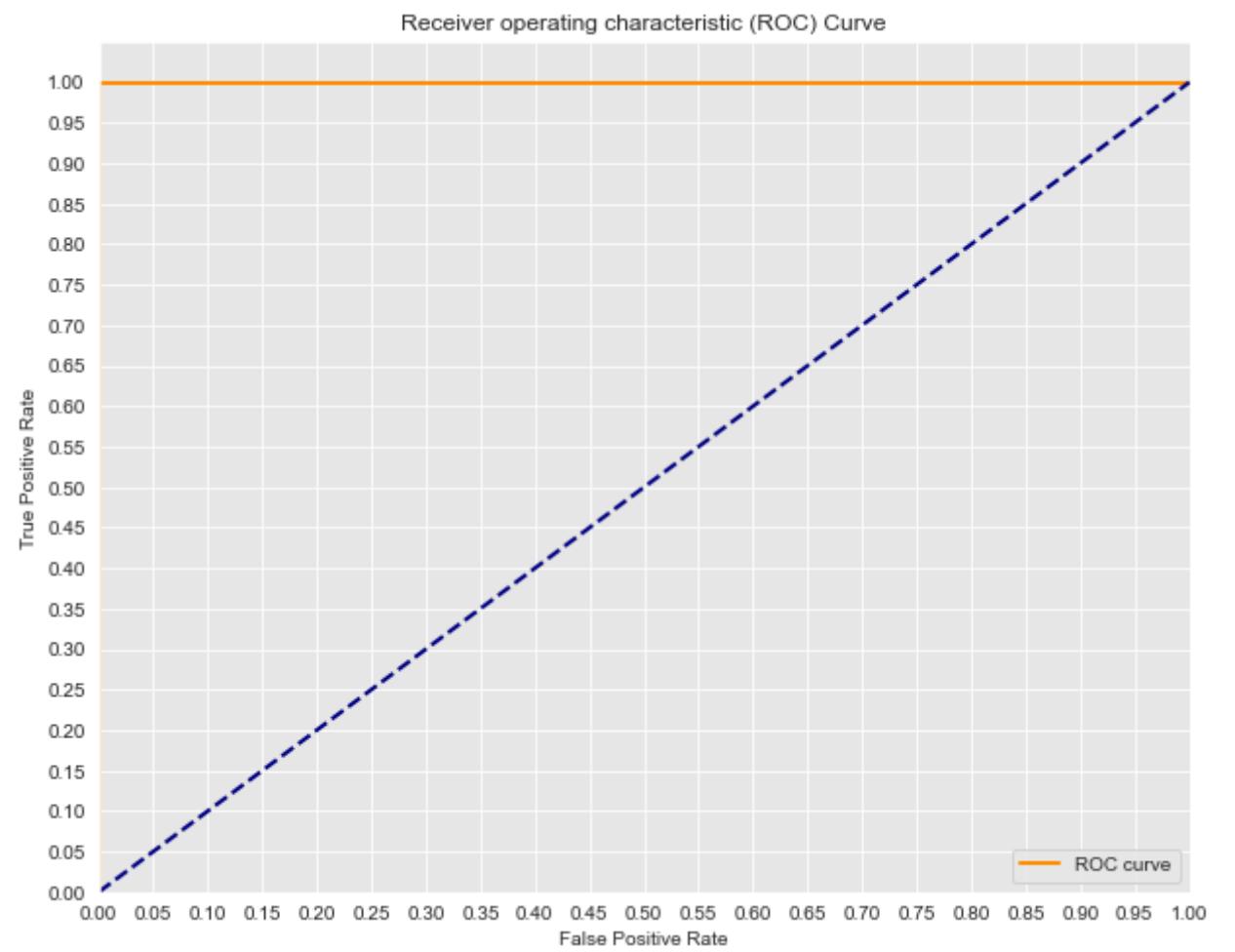
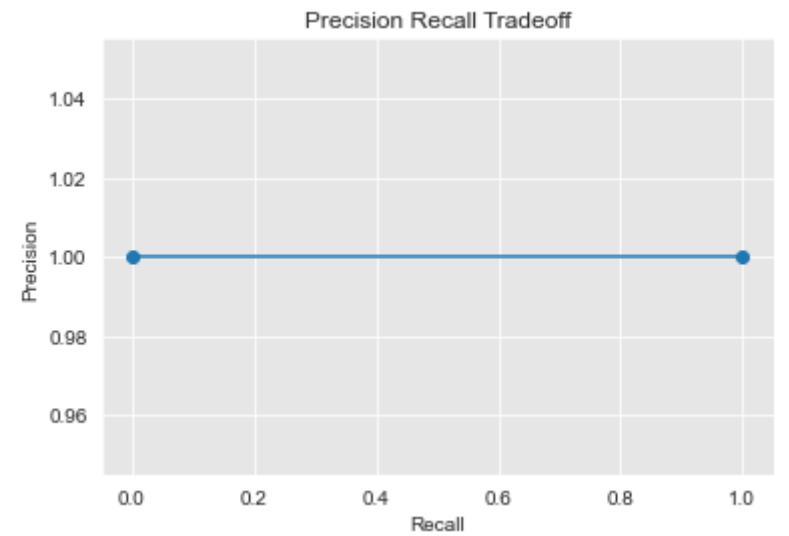
Cross Validated ROC AUC score: 1.0

Class:L
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



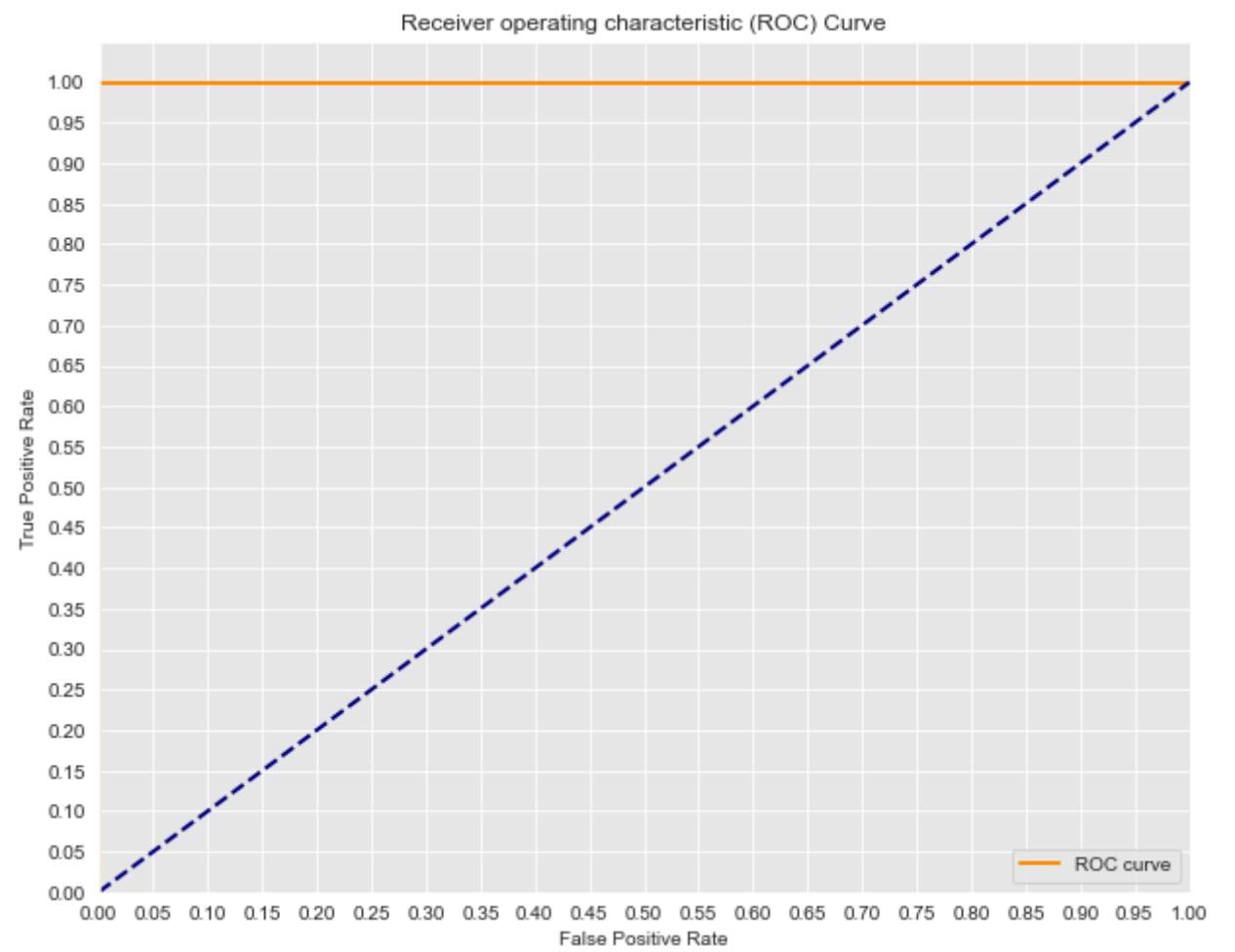
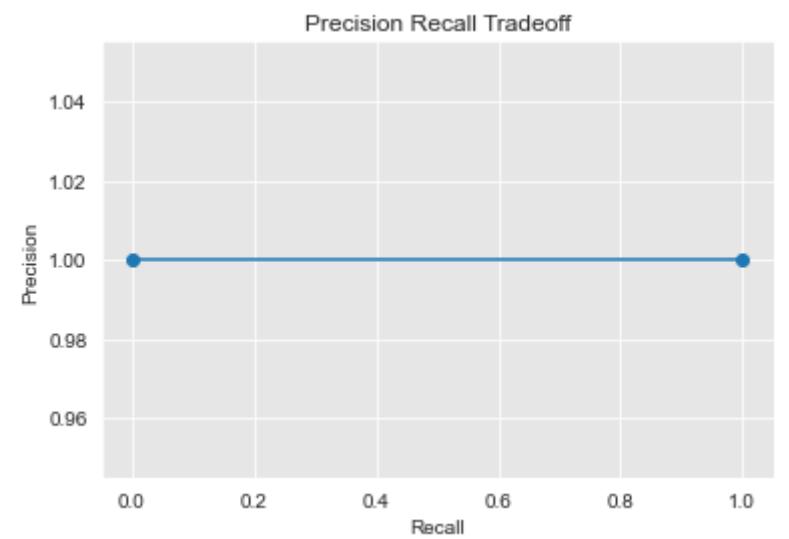
Cross Validated ROC AUC score: 1.0

Class:M
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



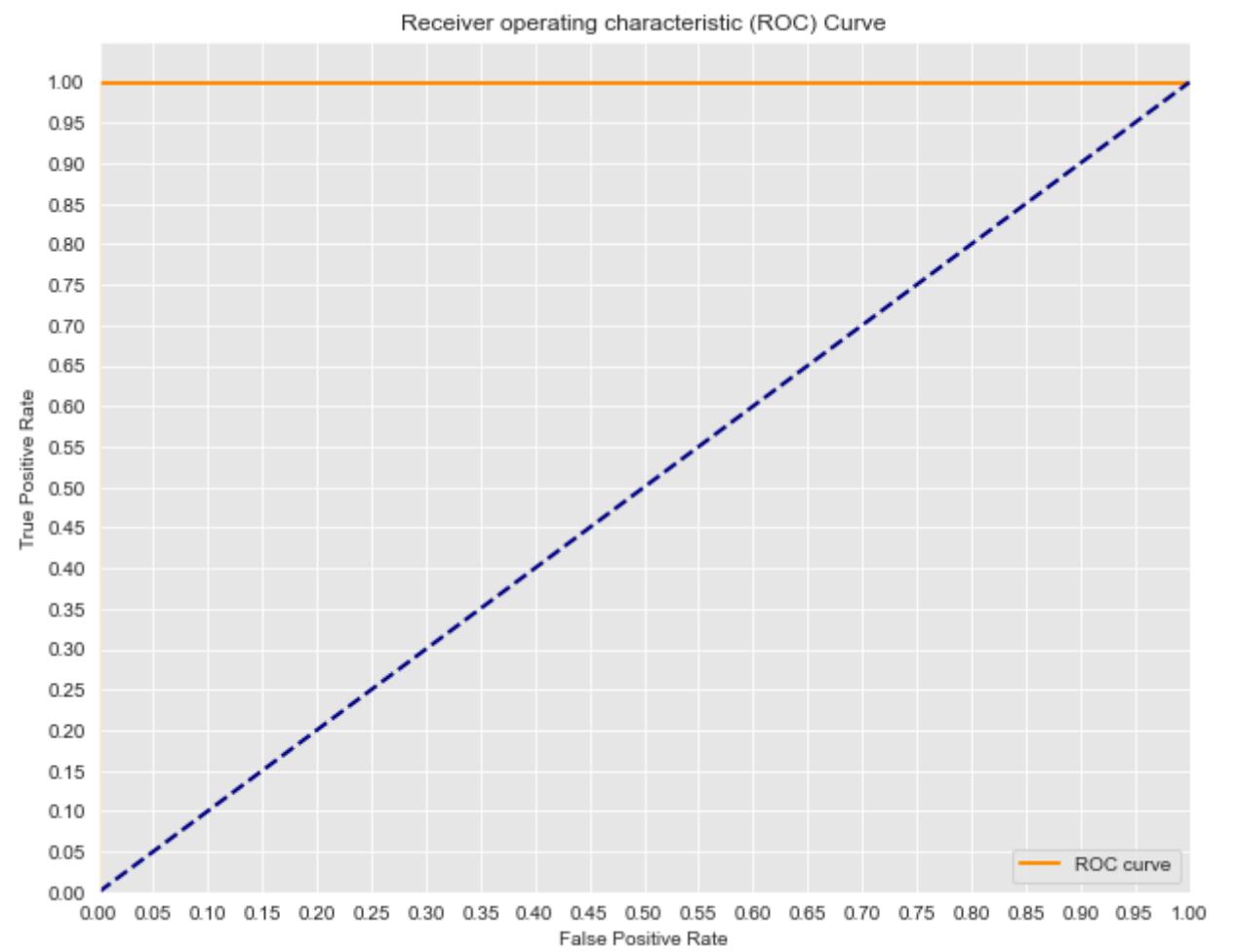
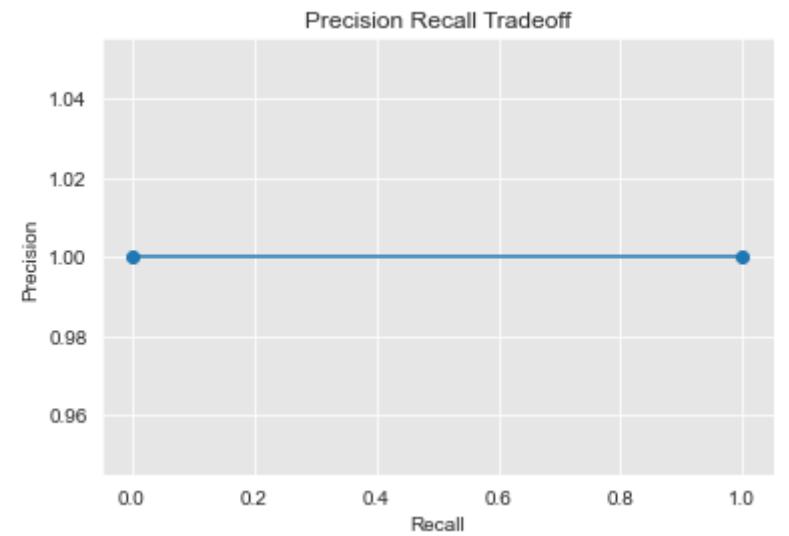
Cross Validated ROC AUC score: 1.0

Class:N
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



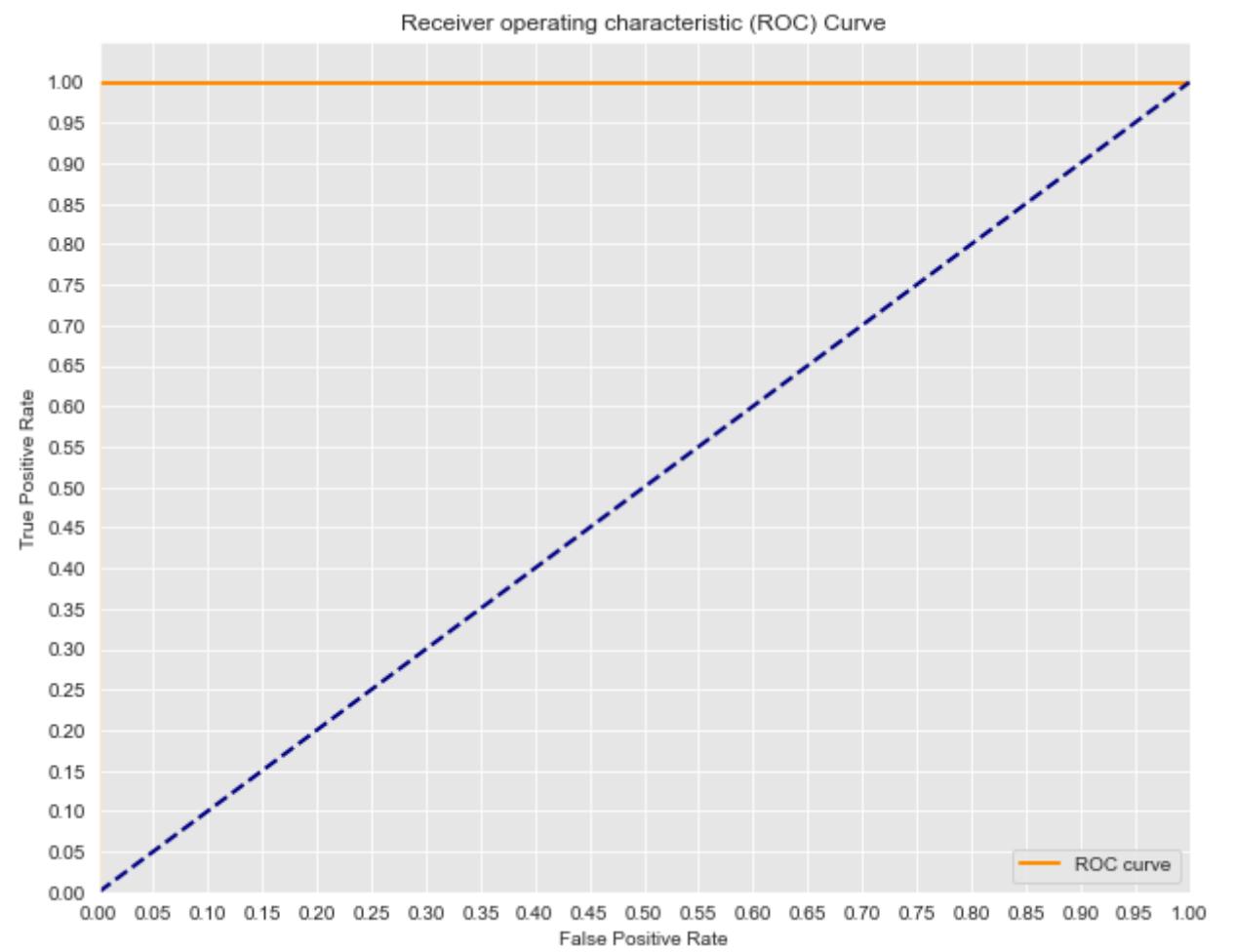
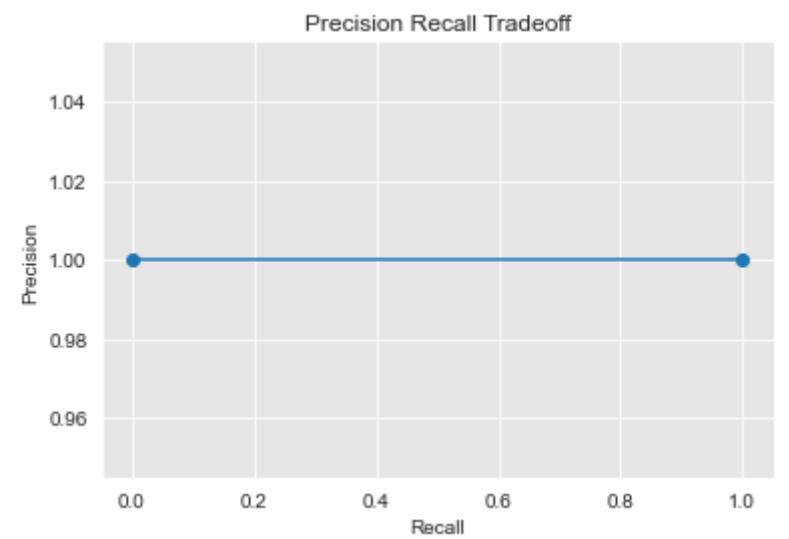
Cross Validated ROC AUC score: 1.0

Class:P
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



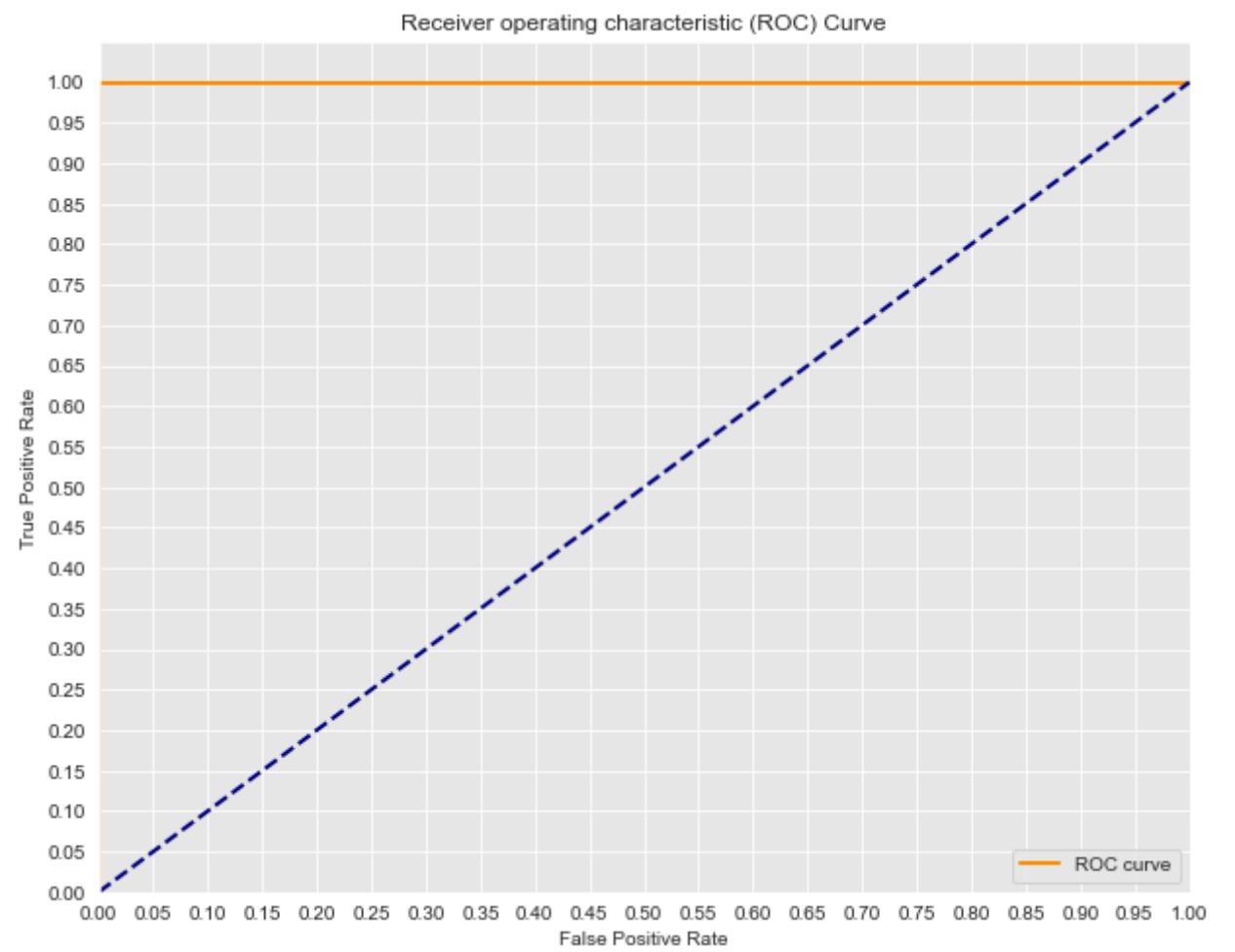
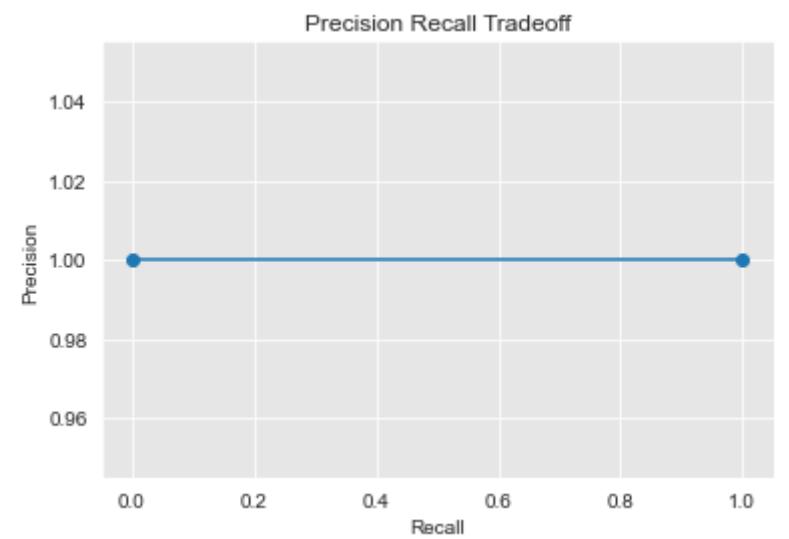
Cross Validated ROC AUC score: 1.0

Class:Q
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



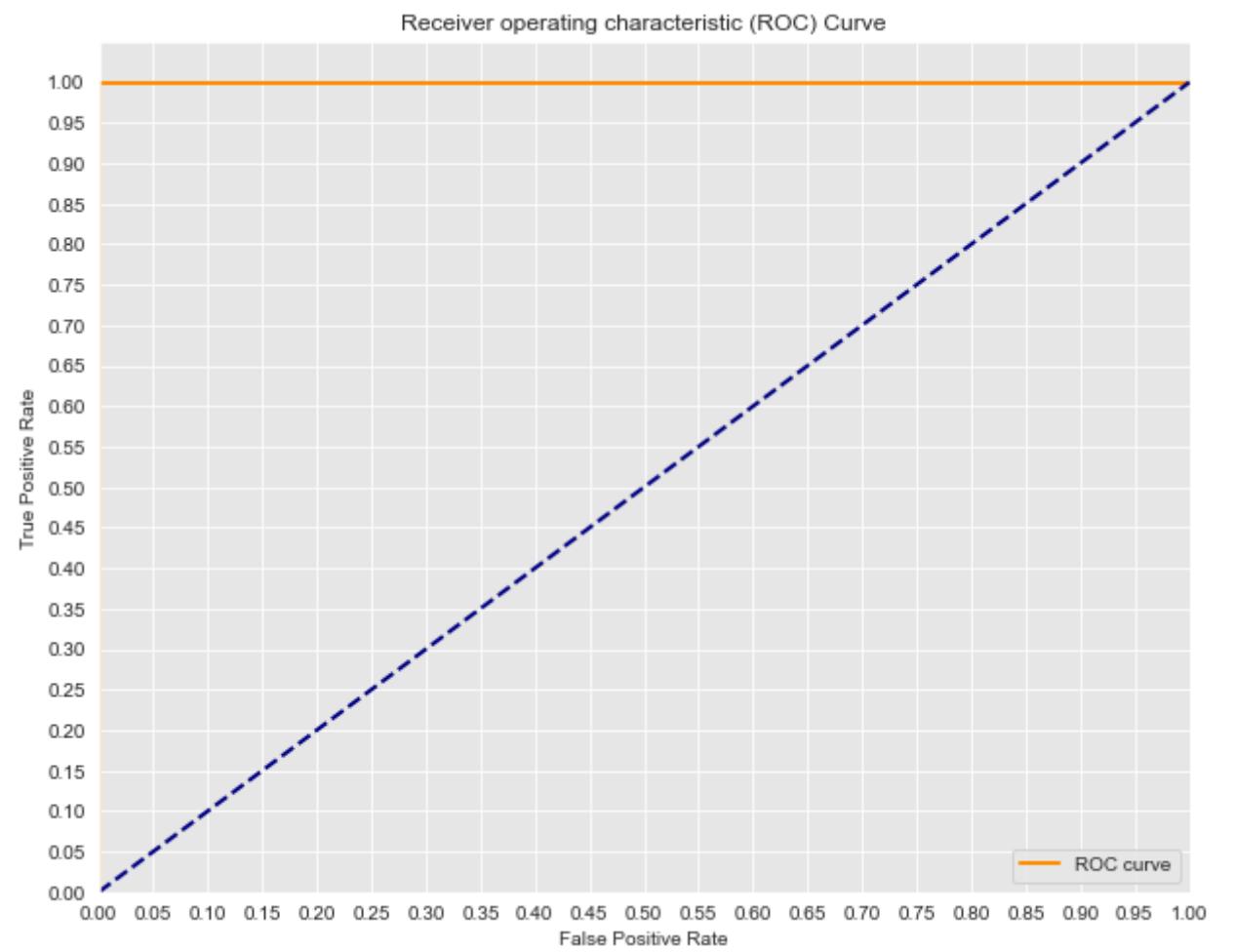
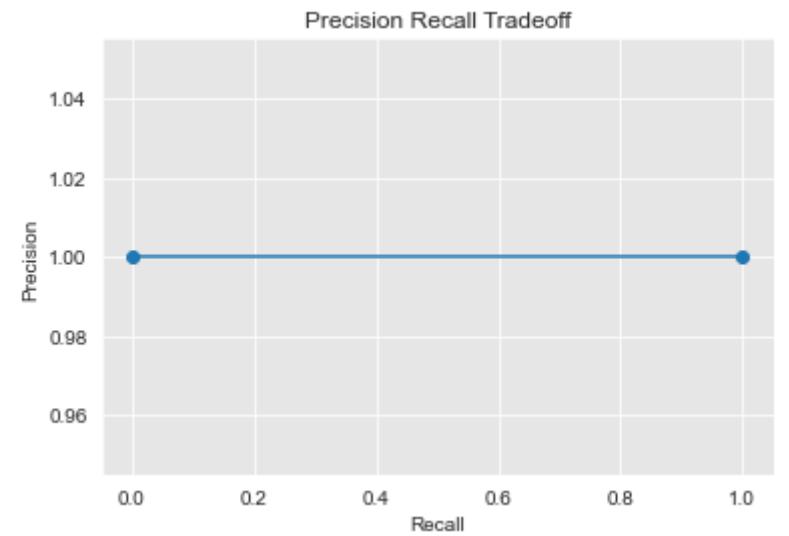
Cross Validated ROC AUC score: 1.0

Class:R
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



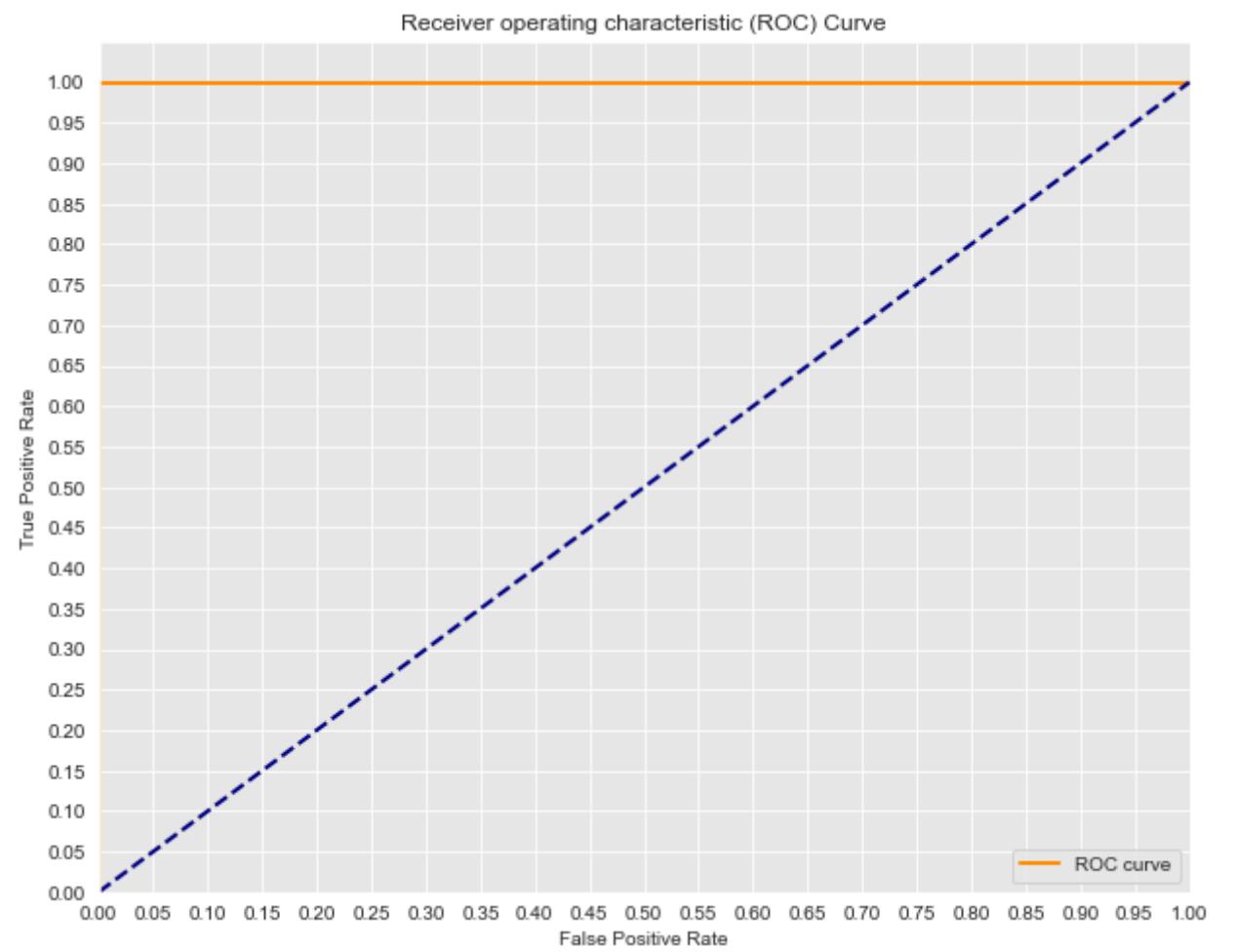
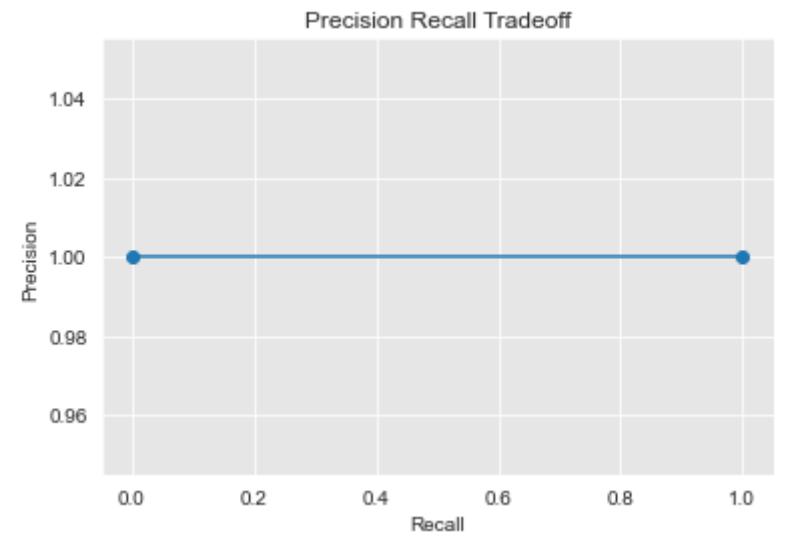
Cross Validated ROC AUC score: 1.0

Class:S
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



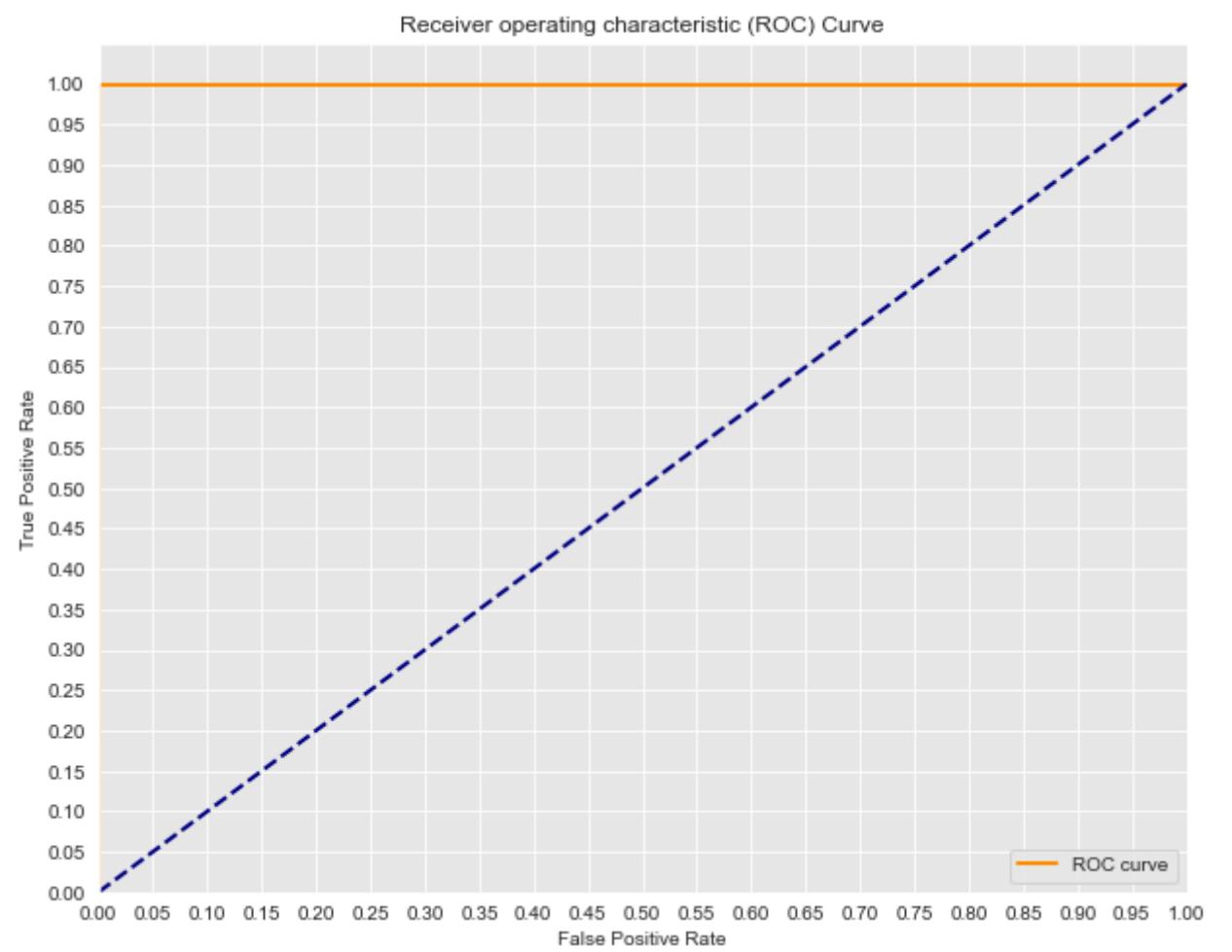
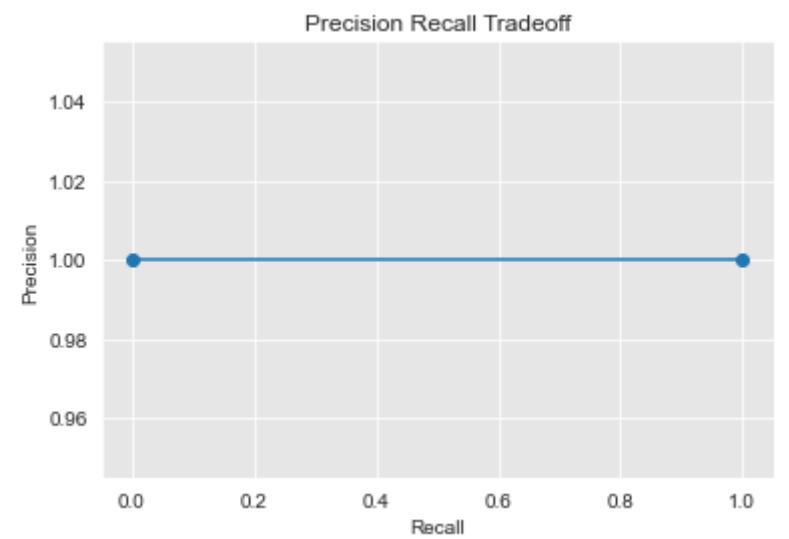
Cross Validated ROC AUC score: 1.0

Class:Stop
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



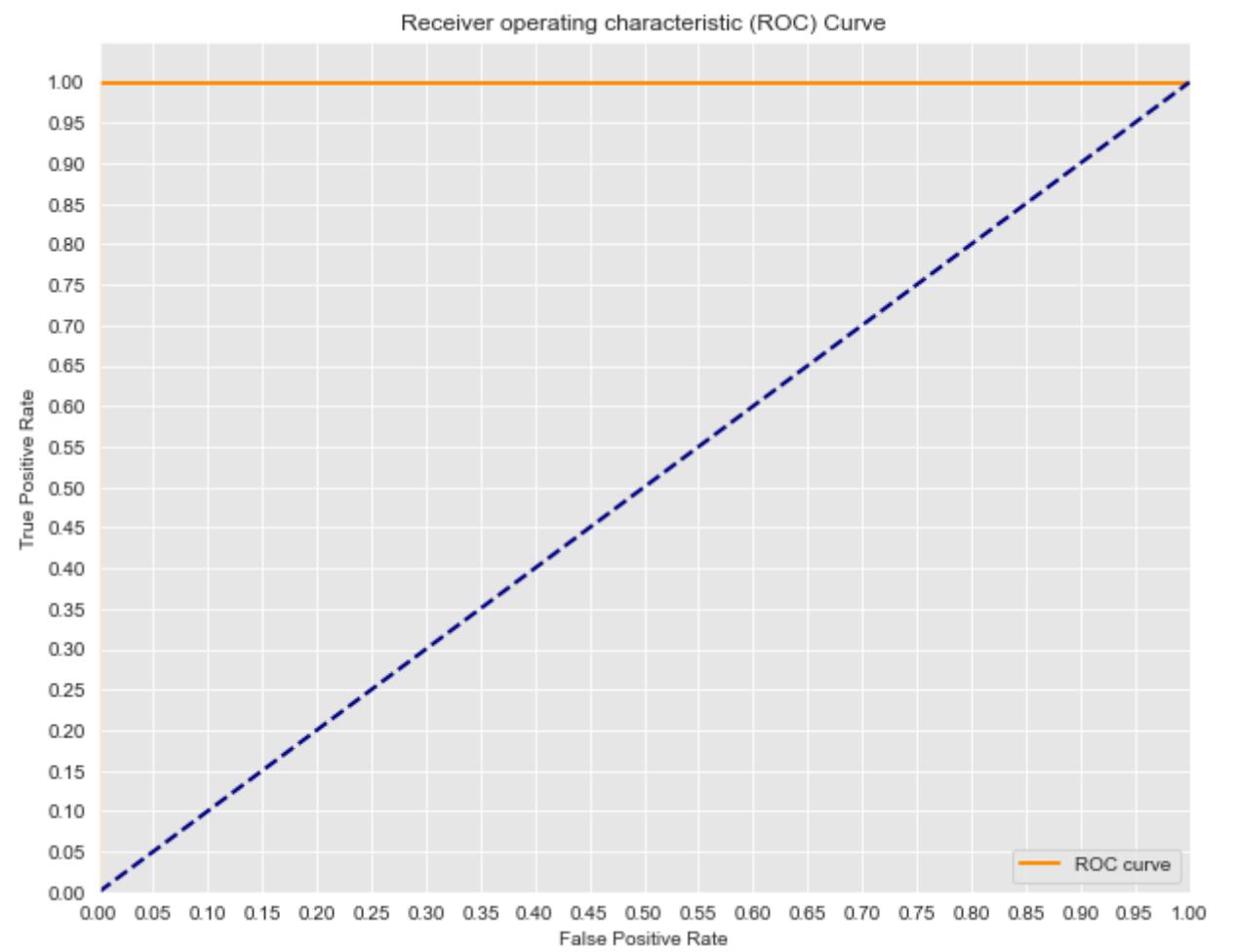
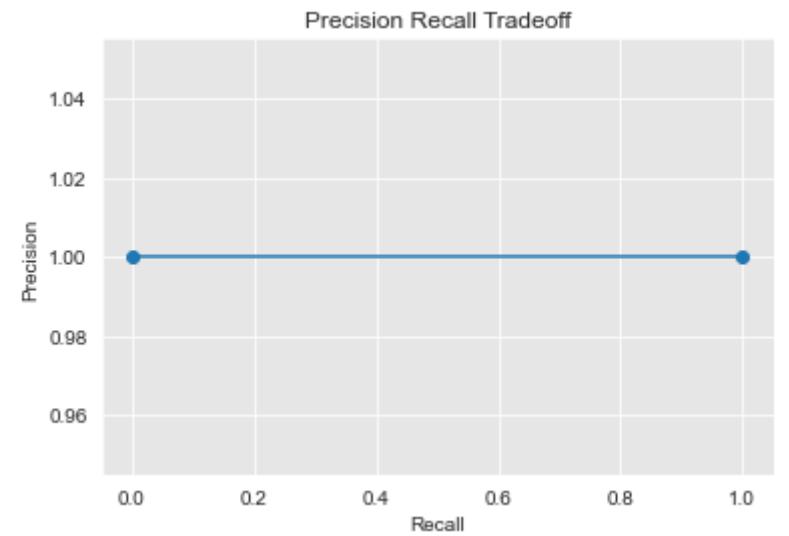
Cross Validated ROC AUC score: 1.0

Class:T
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



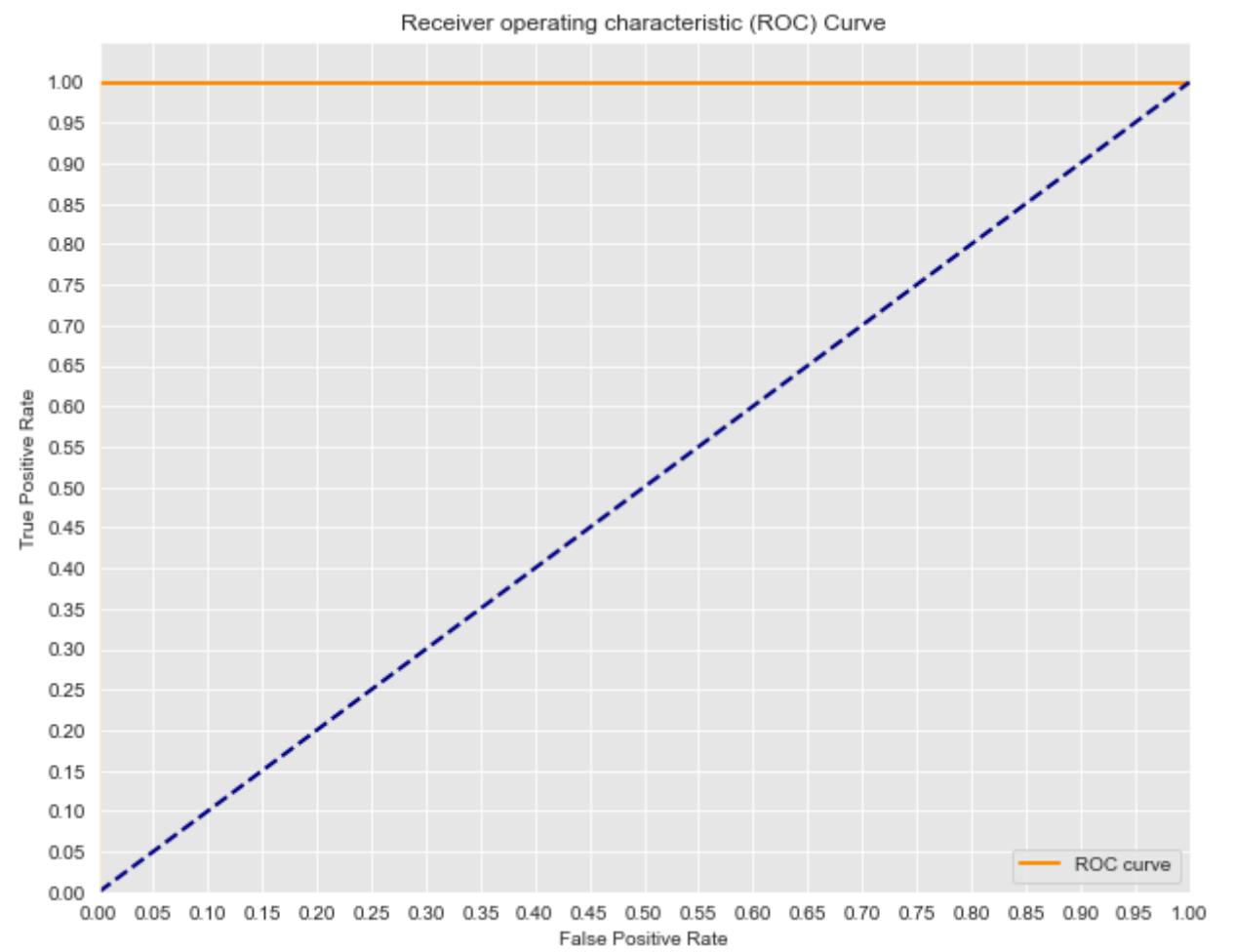
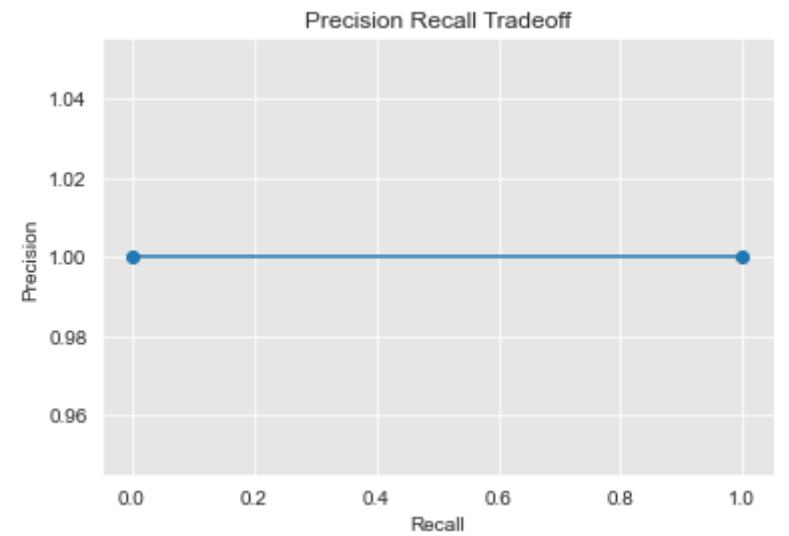
Cross Validated ROC AUC score: 1.0

Class:V
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



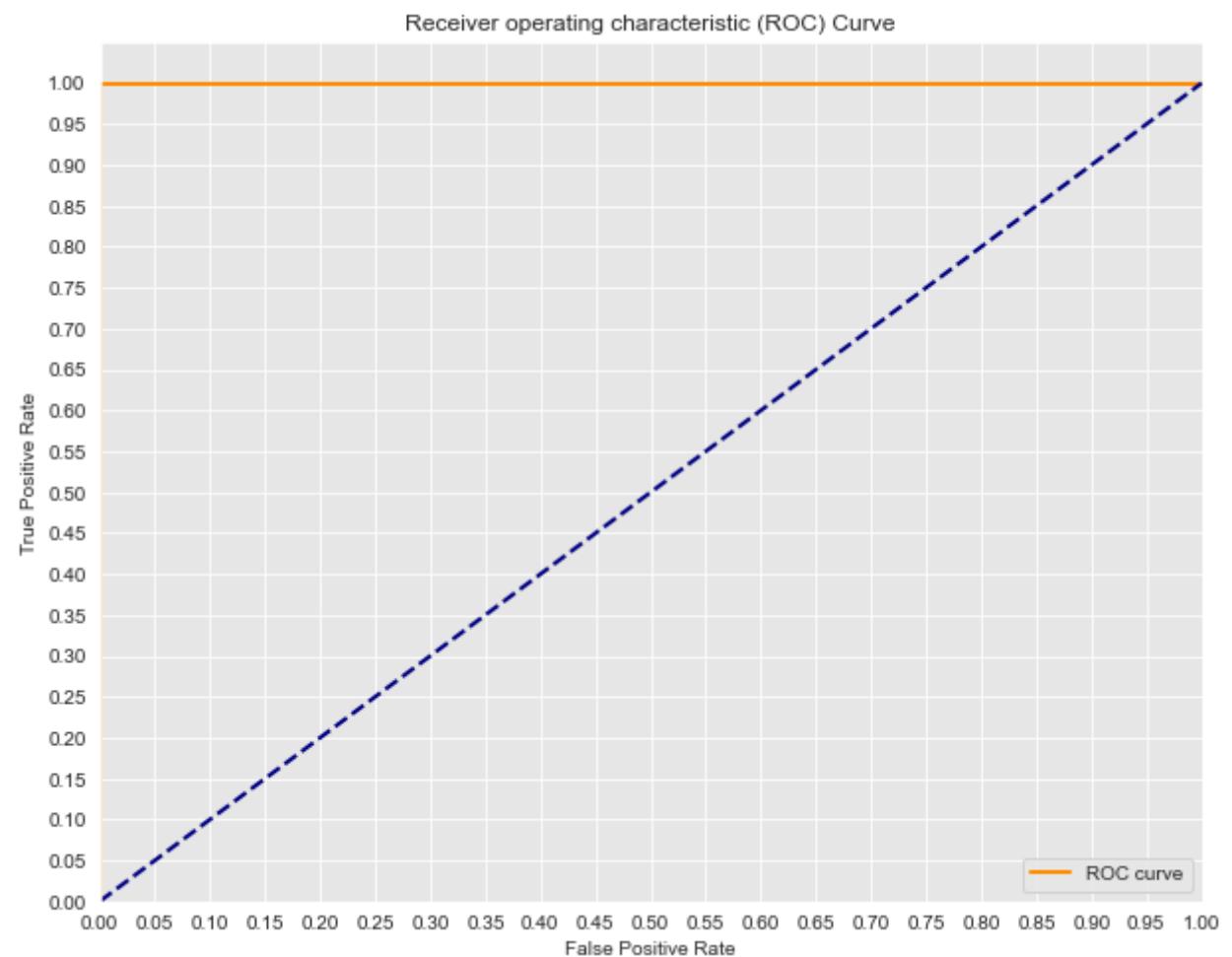
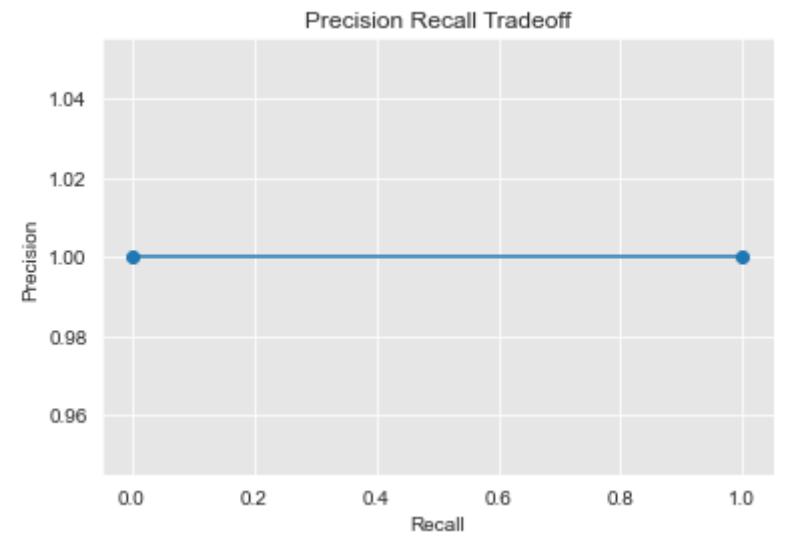
Cross Validated ROC AUC score: 1.0

Class:W
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



Cross Validated ROC AUC score: 1.0

Class:Y
Precision Score: 1.0
Recall Score: 1.0
F1 Score: 1.0
Accuracy Score: 1.0
Specificity Score: 1.0
AUC: 1.0



Cross Validated ROC AUC score: 1.0

The following is an rnn that is under development.

Data Cleaning

Adds a lagged RNA nucleotides column for nucleotide sequence prediction.

```
In [9]: translation_df1['mRNA_Codons_lagged']=translation_df1['mRNA_Codons'].shift(-1)  
translation_df1
```

Out[9]:

	mRNA_Codons	mRNA_Codons_copy	Anticodons	mRNA_Codons_lagged
0	0	A	3	1.0
1	1	C	2	3.0
2	3	U	0	3.0
3	3	U	0	3.0
4	3	U	0	2.0
...
27464	3	U	0	2.0
27465	2	G	1	0.0
27466	0	A	3	3.0
27467	3	U	0	0.0
27468	0	A	3	NaN

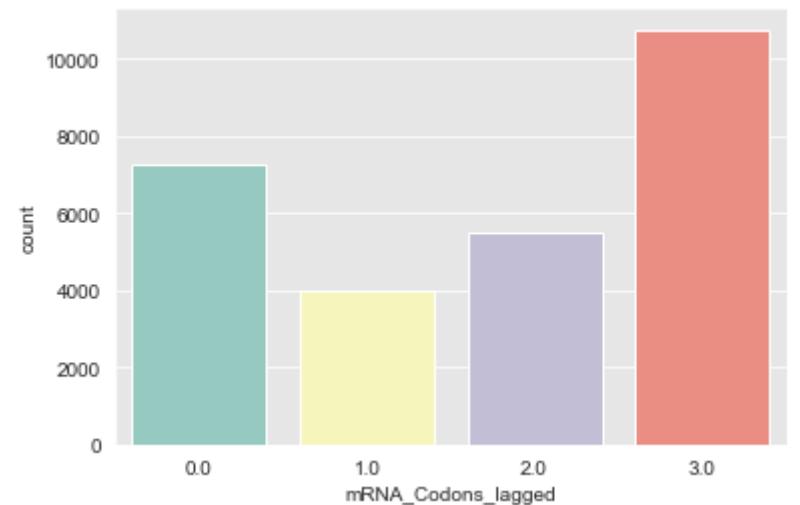
27469 rows × 4 columns

In [10]: `translation_df1['mRNA_Codons_lagged']=translation_df1['mRNA_Codons_lagged'].fillna(0)`

Data Exploration

In [158...]: `sns.countplot(translation_df1['mRNA_Codons_lagged'], palette='Set3')`

Out[158...]: <AxesSubplot:xlabel='mRNA_Codons_lagged', ylabel='count'>



Feature Engineering

In [282...]: `X3=translation_df1['mRNA_Codons']
y3=translation_df1['mRNA_Codons_lagged']`

In [239...]: `from sklearn.preprocessing import MinMaxScaler
X3=X3.values.reshape(-1, 1)`

```
# scaler = MinMaxScaler()
# data_set = scaler.fit_transform(X3)
```

```
In [283...]: X3_train, X3_test, y3_train, y3_test = train_test_split(X3, y3, test_size=0.20, random_state=10)
```

```
In [284...]: X3_train=X3_train.values.reshape(21975,1,1)
```

XOR

```
In [216...]: X3_train=X3_train[:20000]
y3_train=y3_train[:20000]
```

```
In [217...]: X3_train=X3_train.values.reshape(2000,10,1)
```

```
In [218...]: y3_train=y3_train.values.reshape(2000,10,1)
```

```
In [285...]: X3_train = np.asarray(X3_train).astype('float32')
y3_train = np.asarray(y3_train).astype('float32')
```

Deep learning Modeling

```
In [235...]: from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dropout
```

```
In [204...]: from keras.optimizers import Adam
adam=Adam(learning_rate=1e-05,
          beta_1=0.9,
          beta_2=0.999,
          epsilon=1e-07)
```

```
In [286...]: rnn = Sequential()
rnn.add(LSTM(units = 50, activation='tanh', return_sequences = True, input_shape = (1,1)))
rnn.add(Dropout(0.2))
rnn.add(Dense(units = 1, activation='softmax'))
rnn.compile(optimizer = adam, loss='categorical_crossentropy', metrics=["accuracy"])
rnn.summary()
```

Model: "sequential_57"

Layer (type)	Output Shape	Param #
=====		
lstm_60 (LSTM)	(None, 1, 50)	10400
dropout_59 (Dropout)	(None, 1, 50)	0
dense_53 (Dense)	(None, 1, 1)	51
=====		
Total params:	10,451	
Trainable params:	10,451	

```
Non-trainable params: 0
```

```
In [287]: rnn.fit(X3_train, y3_train, epochs = 5, batch_size = 20, verbose=1)
```

```
Epoch 1/5  
1099/1099 [=====] - 2s 2ms/step - loss: 2.0479e-07 - accuracy: 0.1445  
Epoch 2/5  
1099/1099 [=====] - 2s 2ms/step - loss: 2.0479e-07 - accuracy: 0.1445  
Epoch 3/5  
1099/1099 [=====] - 2s 2ms/step - loss: 2.0479e-07 - accuracy: 0.1445  
Epoch 4/5  
1099/1099 [=====] - 2s 2ms/step - loss: 2.0479e-07 - accuracy: 0.1445  
Epoch 5/5  
1099/1099 [=====] - 2s 2ms/step - loss: 2.0479e-07 - accuracy: 0.1445
```

```
Out[287]: <tensorflow.python.keras.callbacks.History at 0x7f9c5cd4e2b0>
```

```
In [288]: predictions=rnn.predict(X3_train, verbose=1)  
predictions
```

```
687/687 [=====] - 1s 793us/step
```

```
Out[288]: array([[[1.]],
```

```
    [[1.]],
```

```
    [[1.]],
```

```
    ...,
```

```
    [[1.]],
```

```
    [[1.]],
```

```
    [[1.]]], dtype=float32)
```

```
In [ ]: scores = model.evaluate(X_test, y_test, batch_size=batch_size, verbose=0)  
print("Model Accuracy: %.2f%%" % (scores[1]*100))
```