project

August 22, 2025

1 Email Spam Detection Project

1.1 Importing Project Tools and Libraries

1.2 Fetching the Dataset

```
  Email No.  the  to  ect  and  for  of    a  you  hou  …  connevey  jay  \
0   Email 1    0   0    1    0    0   0    2    0    0  …         0    0
1   Email 2    8  13   24    6    6   2  102    1   27  …         0    0
2   Email 3    0   0    1    0    0   0    8    0    0  …         0    0
3   Email 4    0   5   22    0    5   1   51    2   10  …         0    0
4   Email 5    7   6   17    1    5   2   57    0    9  …         0    0

   valued  lay  infrastructure  military  allowing  ff  dry  Prediction
0       0    0               0         0         0   0    0           0
1       0    0               0         0         0   1    0           0
2       0    0               0         0         0   0    0           0
3       0    0               0         0         0   0    0           0
4       0    0               0         0         0   1    0           0

[5 rows x 3002 columns]
```

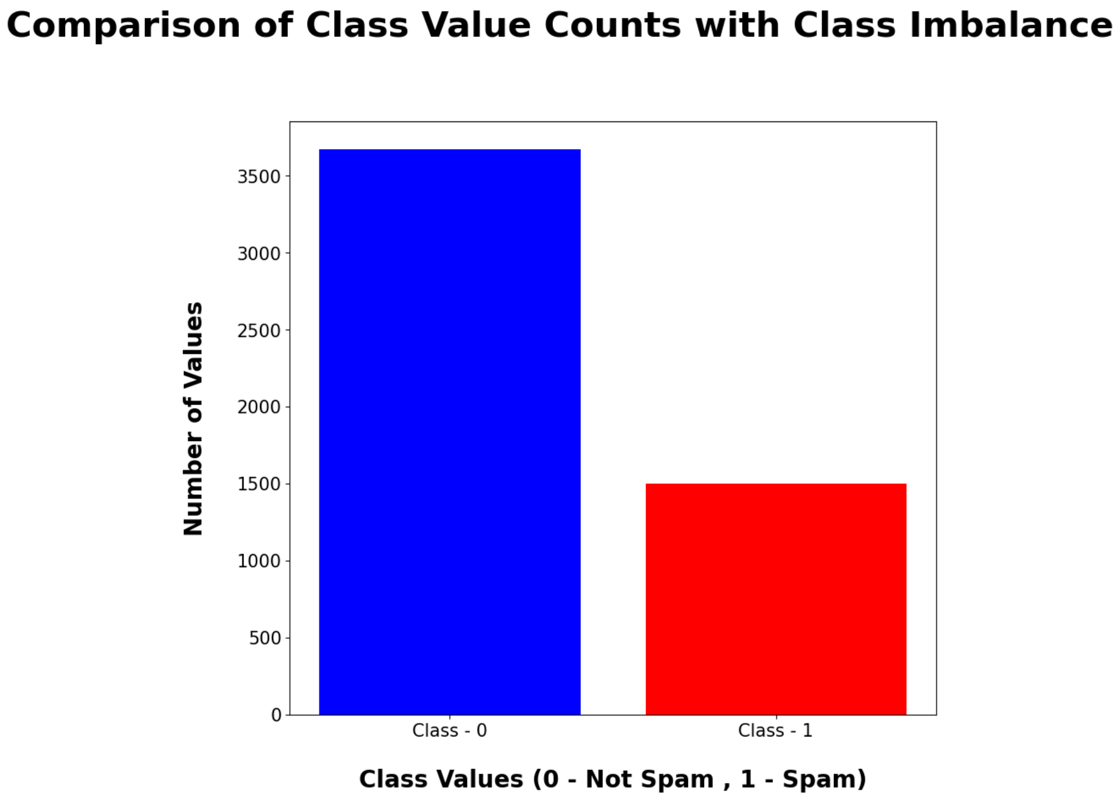
1.3 Analyzing the Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3002 entries, Email No. to Prediction
dtypes: int64(3001), object(1)
memory usage: 118.5+ MB
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5172 entries, 0 to 5171
Columns: 3001 entries, the to Prediction
dtypes: int64(3001)
memory usage: 118.4 MB
```

```
(0.7099767981438515, 0.2900232018561485)
```

1.4 Visualizing the Class Imbalance present in the Dataset

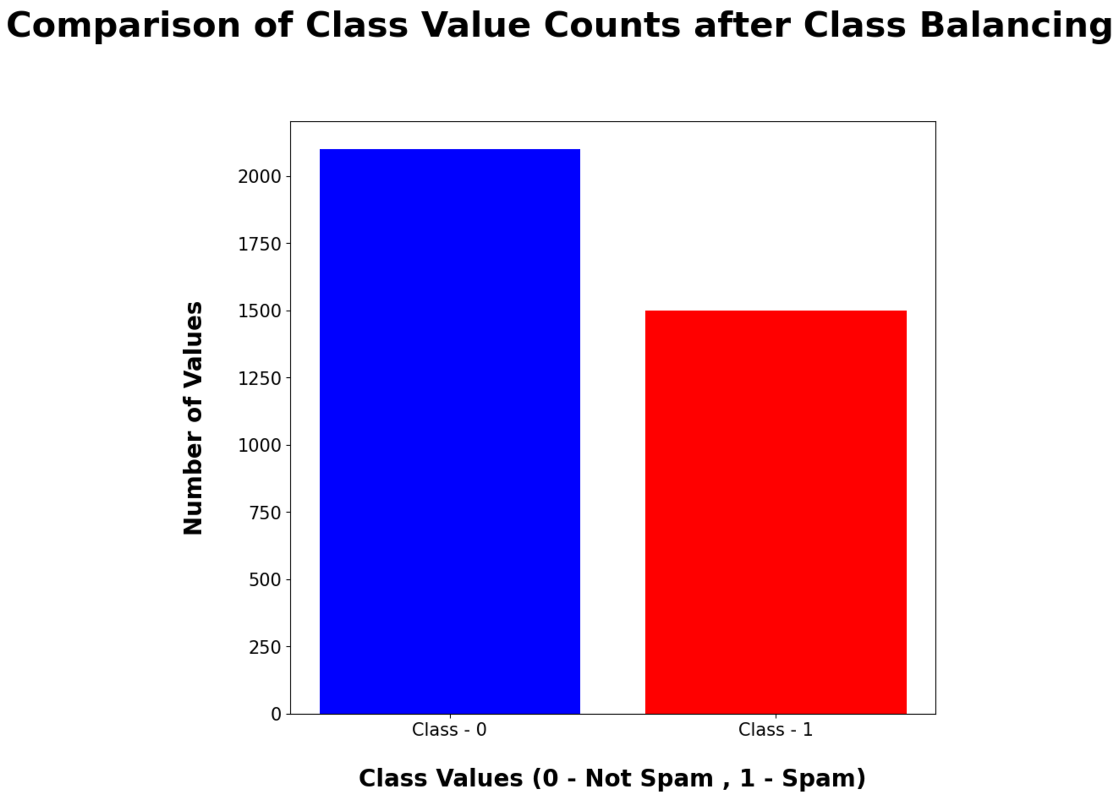


1.5 Improving (decreasing) the Class Imabalance in the Dataset

```
<class 'pandas.core.frame.DataFrame'>
Index: 3600 entries, 0 to 5170
Columns: 3001 entries, the to Prediction
dtypes: int64(3001)
memory usage: 82.5 MB
```

```
(0.5833333333333334, 0.4166666666666667)
```

1.6 Visualizing the Class Balances after Improvements



1.7 Analyzing the Correlation

	the	to	ect	and	for	of	a \
the	1.000000	0.861043	0.349717	0.850516	0.813911	0.817475	0.794985
to	0.861043	1.000000	0.394282	0.832428	0.815526	0.779931	0.915871
ect	0.349717	0.394282	1.000000	0.280116	0.384158	0.175051	0.414901
and	0.850516	0.832428	0.280116	1.000000	0.772129	0.830381	0.806145
for	0.813911	0.815526	0.384158	0.772129	1.000000	0.716931	0.769736

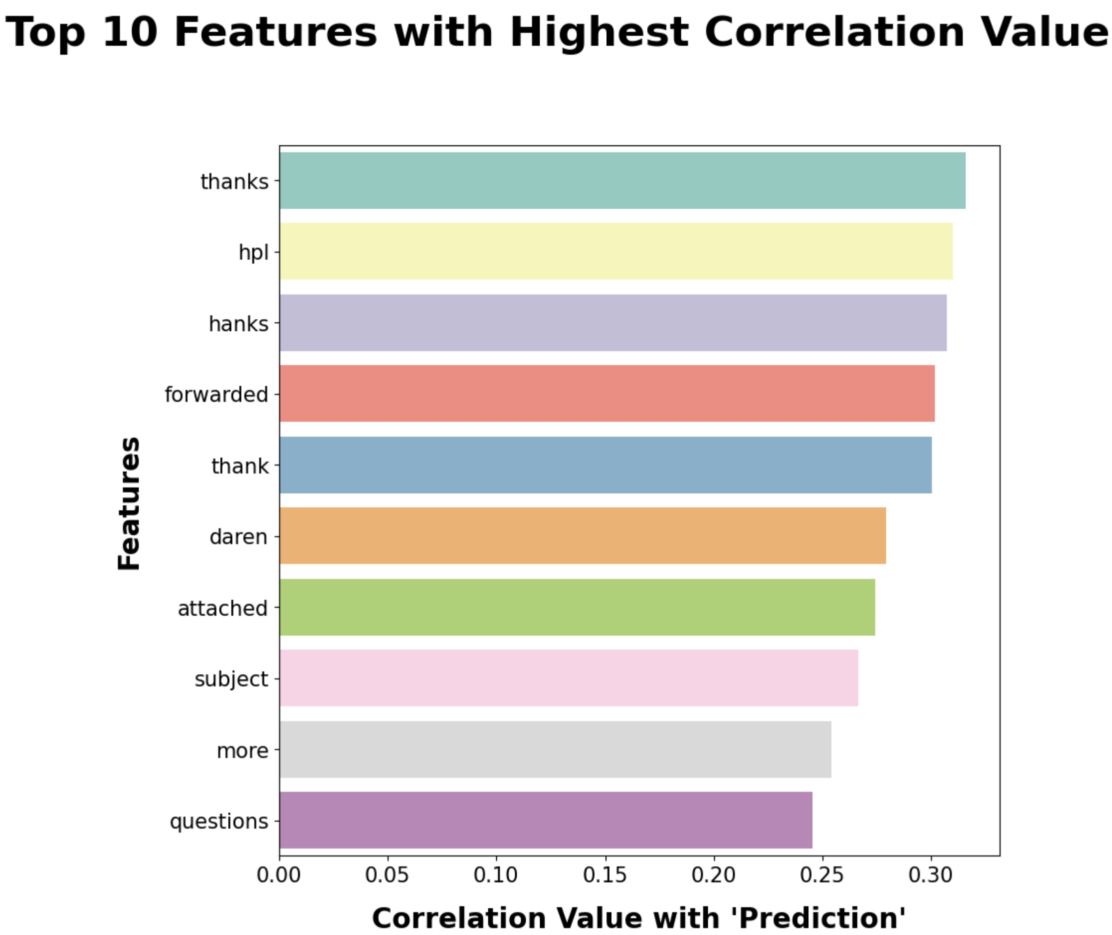
	you	hou	in	…	connevey	jay	valued \
the	0.488345	0.312697	0.859270	…	0.010626	0.093534	0.268920
to	0.512931	0.359153	0.895648	…	0.015148	0.095719	0.274160
ect	0.152610	0.982827	0.297639	…	0.141289	0.049312	0.047024
and	0.470454	0.237936	0.878319	…	0.004673	0.182575	0.316148
for	0.500822	0.346128	0.786190	…	0.024743	0.093076	0.278534

	lay	infrastructure	military	allowing	ff	dry \
the	0.217945	0.103570	0.135583	0.122523	0.339213	0.035825
to	0.252879	0.099826	0.089169	0.123699	0.404978	0.082645
ect	0.066638	0.003271	-0.010459	0.002749	0.137480	0.002560

```
and  0.236568        0.163353  0.074078  0.129192  0.393962  0.033334
for  0.219834        0.147242  0.067918  0.116457  0.313685  0.033095

     Prediction
the   -0.001691
to     0.067904
ect   -0.163373
and    0.125875
for   -0.013451

[5 rows x 3001 columns]
```

1.8 Visualizing the Top 10 highest Correlation Value Features (w.r.t "Prediction")

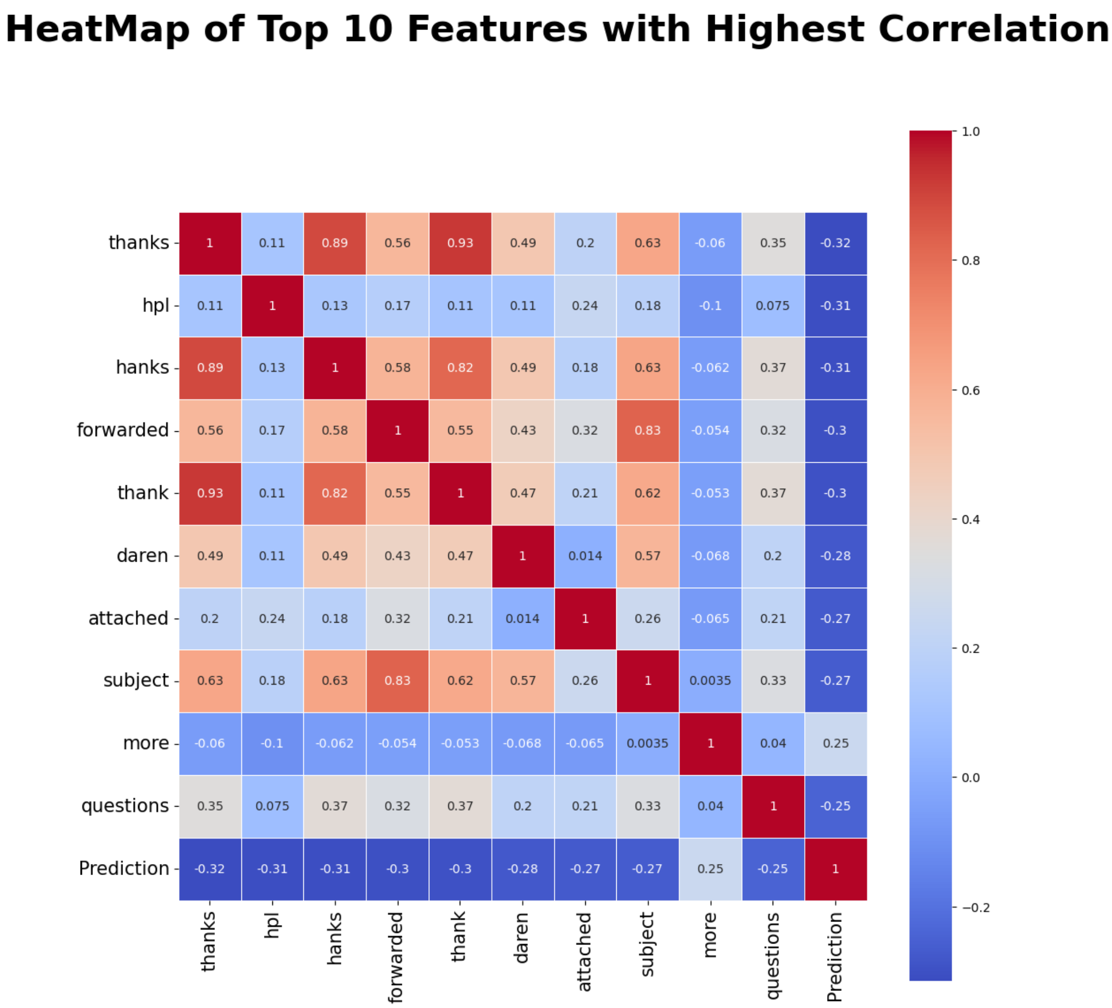
```
[['thanks', np.float64(0.3159054984041723)],
 ['hpl', np.float64(0.3100339769414807)],
 ['hanks', np.float64(0.3072310178680418)],
 ['forwarded', np.float64(0.30157291141236847)],
 ['thank', np.float64(0.30023069171555045)],
 ['daren', np.float64(0.27911374237570663)],
 ['attached', np.float64(0.2743678951290051)],
 ['subject', np.float64(0.26647837827651544)],
 ['more', np.float64(0.25404028032380477)],
 ['questions', np.float64(0.24528306903766356)]]
```



1.9 Creating a HeatMap of the Top 10 Best Features

	thanks	hpl	hanks	forwarded	thank	daren	attached	subject	more	\
0	0	0	0	0	0	0	0	0	0	
1	1	0	1	3	1	3	1	3	0	
2	0	0	0	0	0	0	0	0	0	
3	1	0	1	2	1	2	0	3	0	
4	1	0	1	2	1	1	0	2	0	

	questions	Prediction
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0



1.10 Splitting the dataset into independent and dependent variables

```
(   the  to  ect  and  for  of    a  you  hou  in  …  enhancements  connevey  \
 0    0   0    1    0    0   0    2    0    0   0  …             0         0
 1    8  13   24    6    6   2  102    1   27  18  …             0         0
 2    0   0    1    0    0   0    8    0    0   4  …             0         0
 3    0   5   22    0    5   1   51    2   10   1  …             0         0
 4    7   6   17    1    5   2   57    0    9   3  …             0         0

    jay  valued  lay  infrastructure  military  allowing  ff  dry
 0    0       0    0               0         0         0   0    0
 1    0       0    0               0         0         0   1    0
 2    0       0    0               0         0         0   0    0
 3    0       0    0               0         0         0   0    0
 4    0       0    0               0         0         0   1    0
```

```
 [5 rows x 3000 columns],
    Prediction
 0           0
 1           0
 2           0
 3           0
 4           0)
```

1.11 Using sklearn pipeline for Data Preprocessing

1.12 Dividing the Dataset into Training and Testing sets

1.13 Training Machine Learning Models on the Training Set

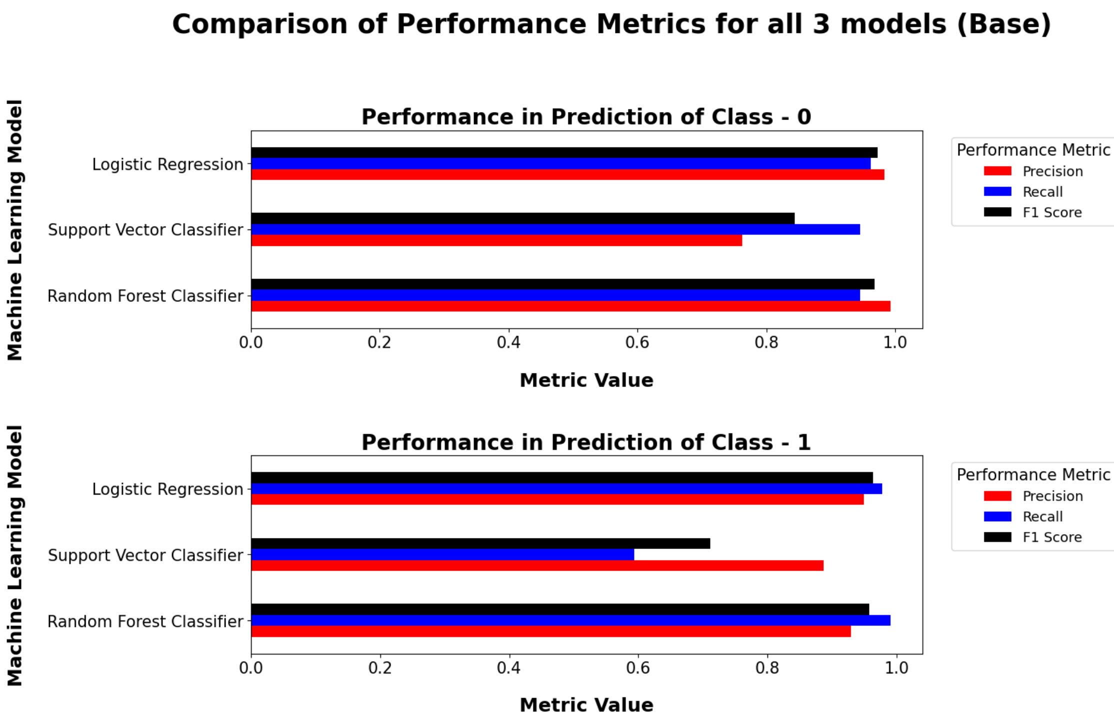
1.14 Creating a Single Function for calculating Performance Metrics for each Model

1.15 Analyzing Performance Metrics for each Model

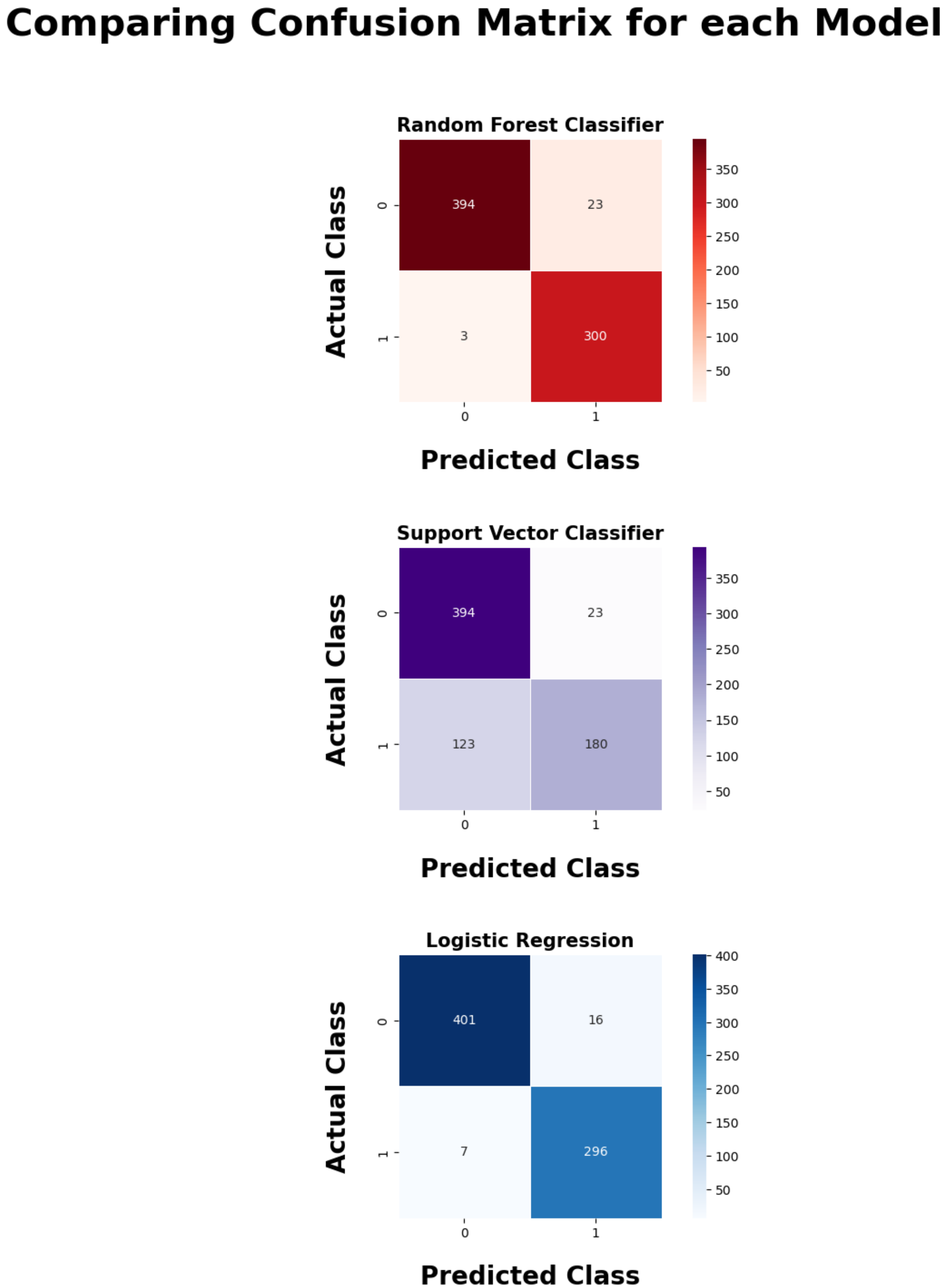
```
Performance Metrics for Random Forest Classifier:
For Class = 0
Precision: 0.992
Recall: 0.945
F1 Score: 0.968

For Class = 1
Precision: 0.929
Recall: 0.99
F1 Score: 0.958

Mean Accuracy: 0.964

Performance Metrics for Support Vector Classifier:
For Class = 0
Precision: 0.762
Recall: 0.945
F1 Score: 0.844

For Class = 1
Precision: 0.887
Recall: 0.594
F1 Score: 0.711

Mean Accuracy: 0.797

Performance Metrics for Logistic Regression:
For Class = 0
```

```
Precision: 0.983
Recall: 0.962
F1 Score: 0.972

For Class = 1
Precision: 0.949
Recall: 0.977
F1 Score: 0.963

Mean Accuracy: 0.968
```

1.16 Visualizing the Performance Metrics for each Model using BarPlot



1.17 Analyzing the Confusion Matrix for each Model using HeatMap

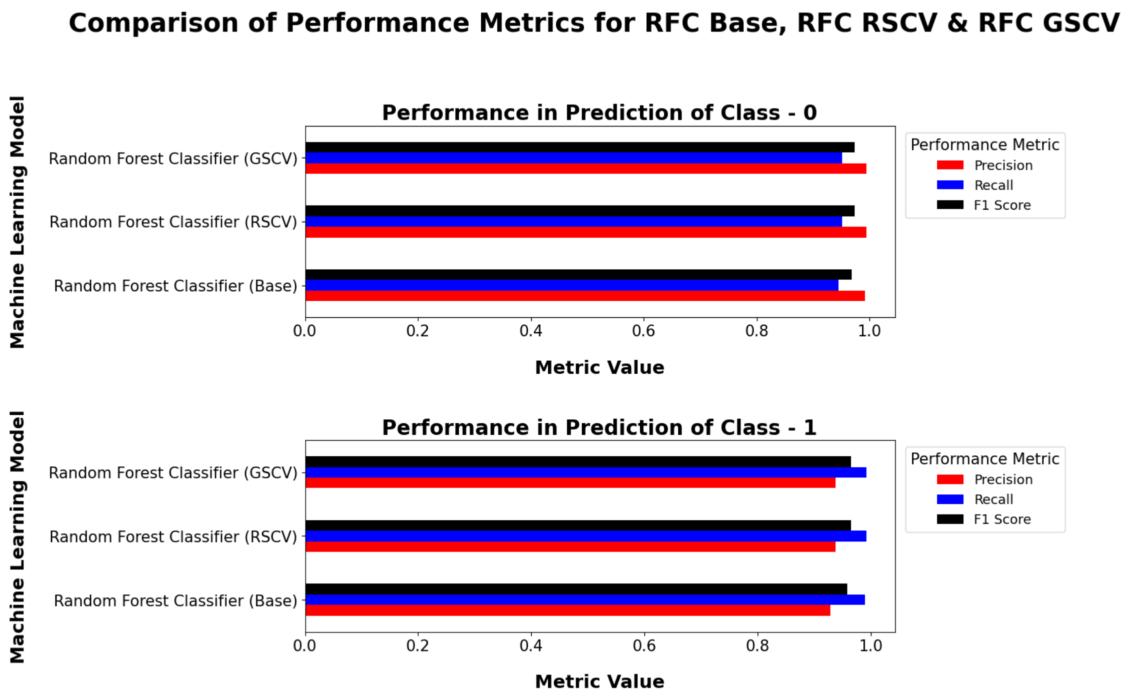


1.18 Tuning Hyperparameters of Random Forest Classifier using Randomized Search Cross Validation

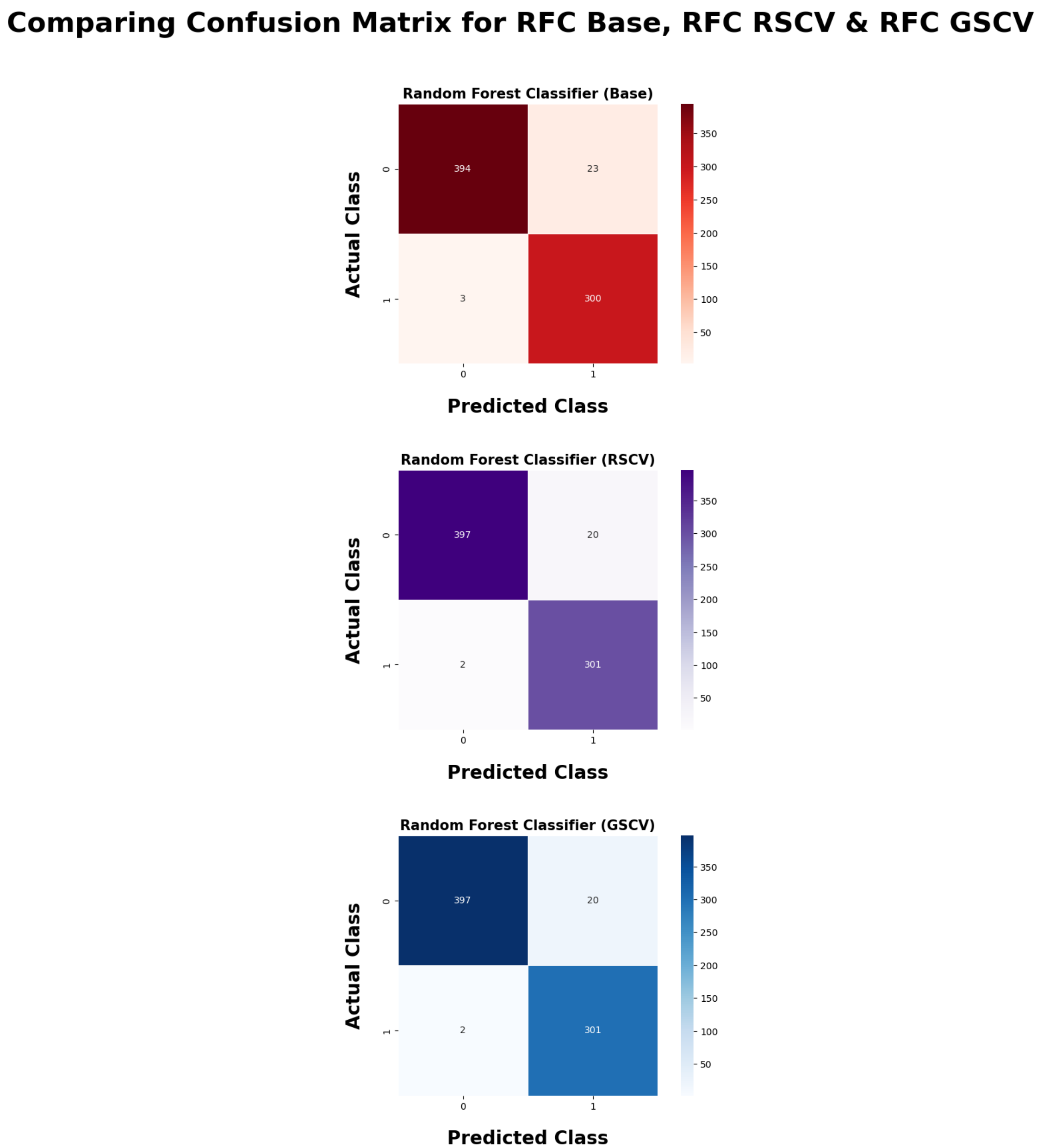
```
{'model__n_estimators': 190,
 'model__min_samples_split': 3,
 'model__min_samples_leaf': 1,
 'model__max_features': 'sqrt',
 'model__max_depth': None}
```

1.19 Tuning hyperparameters of Random Forest Classifier using Grid Search Cross Validation

1.20 Visualizing and Comparing Performance Metrics for all 3 versions of Random Forest Classifier (Base, RSCV Tuned and GSCV Tuned)



1.21 Visualizing and Comparing Confusion Matrix for all 3 versions of Random Forest Classifier (Base, RSCV Tuned and GSCV Tuned)

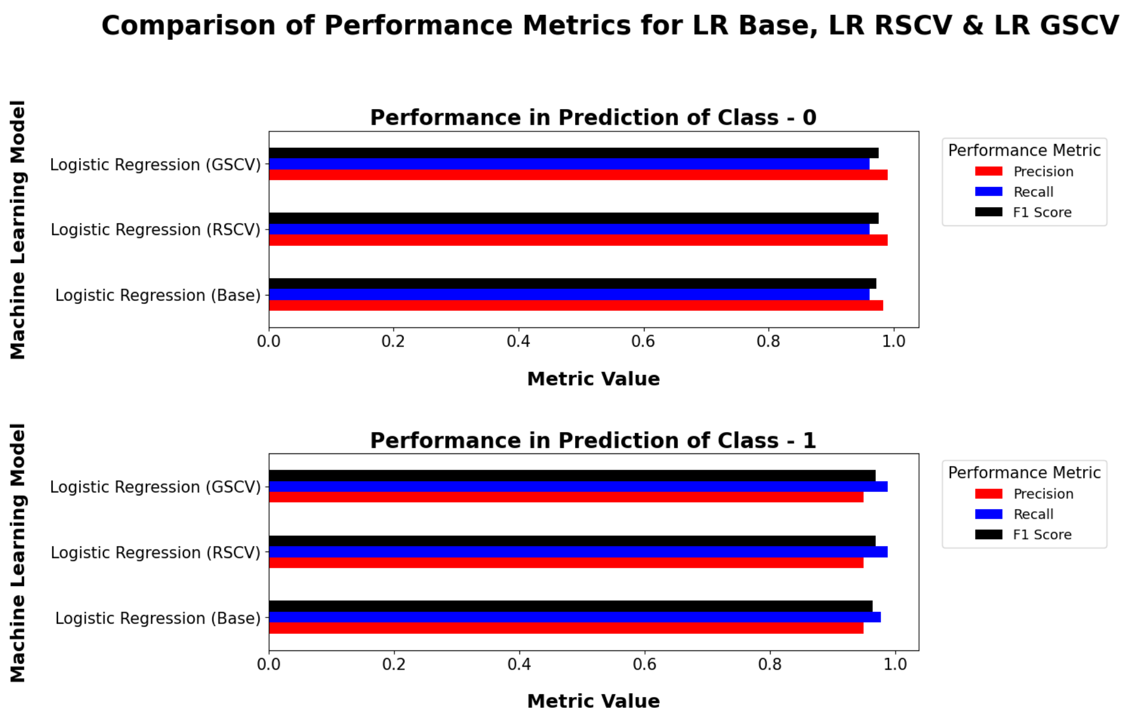


1.22 Tuning hyperparameters of Logistic Regression using Randomized Search Cross Validation

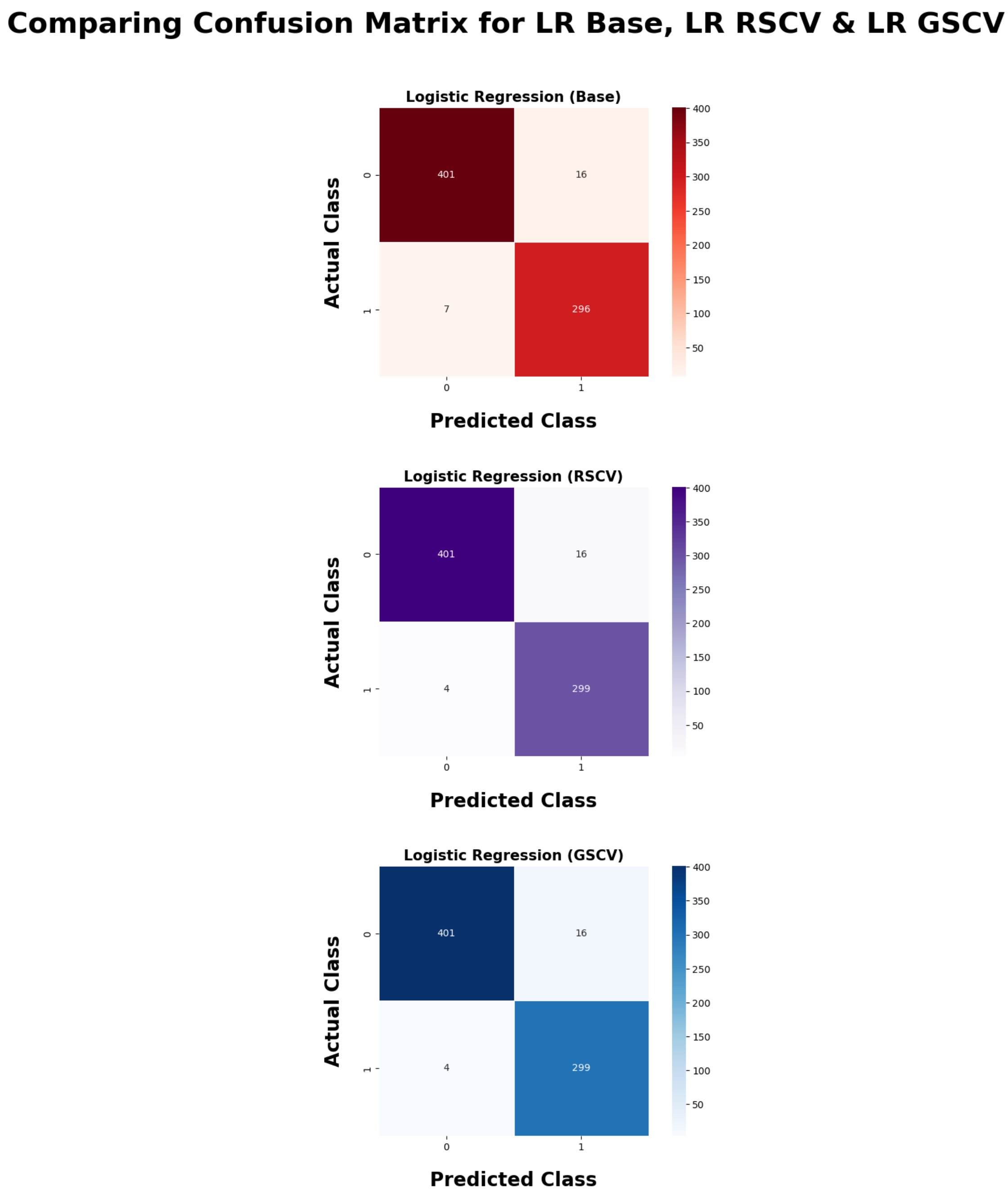
1.23 Tuning hyperparameters of Logistic Regression using Grid Search Cross Validation

```
{'model__C': 0.1,
 'model__max_iter': 100,
 'model__penalty': 'l2',
 'model__solver': 'liblinear'}
```

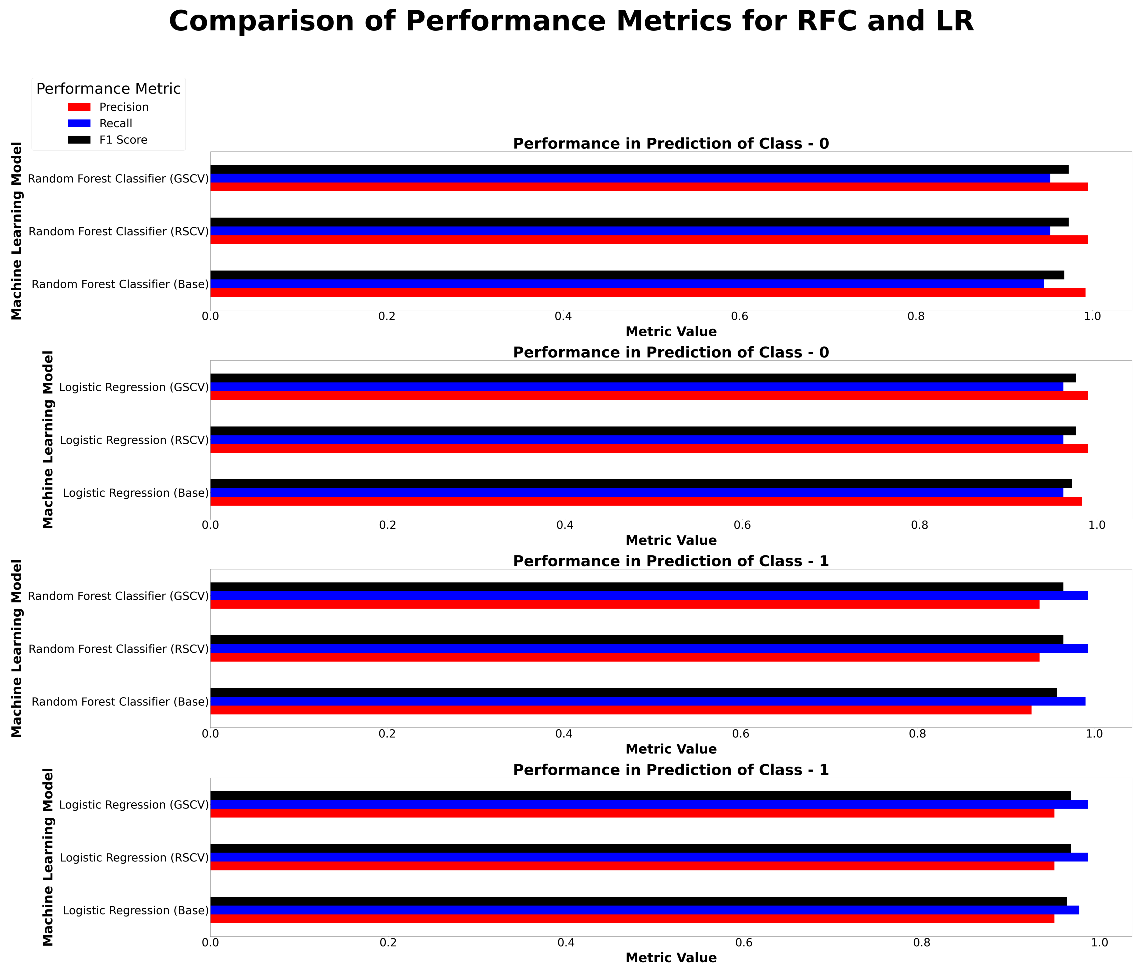
1.24 Visualizing and Comparing Performance Metrics for all 3 versions of Logistic Regression (Base, RSCV Tuned and GSCV Tuned)



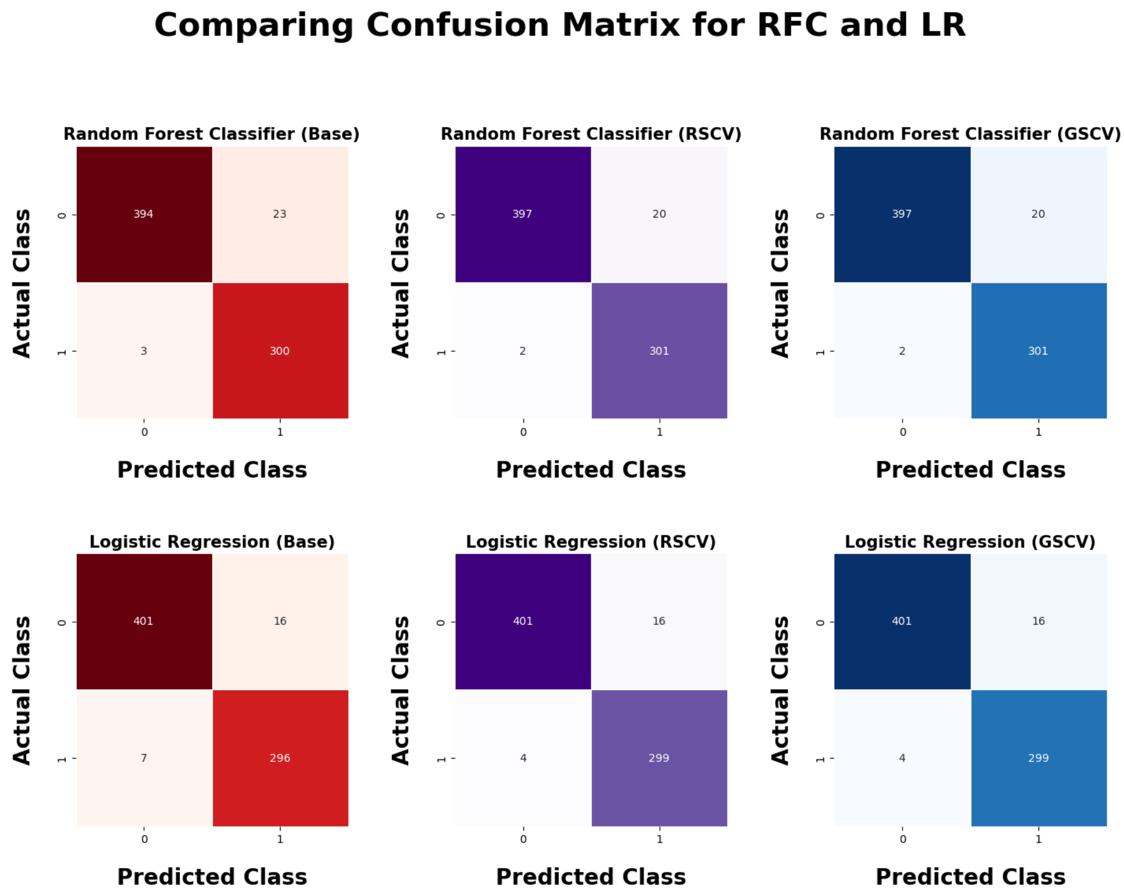
1.25 Visualizing and Comparing Confusion Matrix for all 3 versions of Logistic Regression (Base, RSCV Tuned and GSCV Tuned)



1.26 Summarizing Performance Metrics for all 3 versions of Random Forest Classifier and Logistic Regression



1.27 Summarizing Confusion Matrix for all 3 versions of Random Forest Classifier and Logistic Regression



1.28 Choosing the Best Overall Final Model = GSCV Tuned Logistic Regression

```
['best_model_tuned.joblib']
```