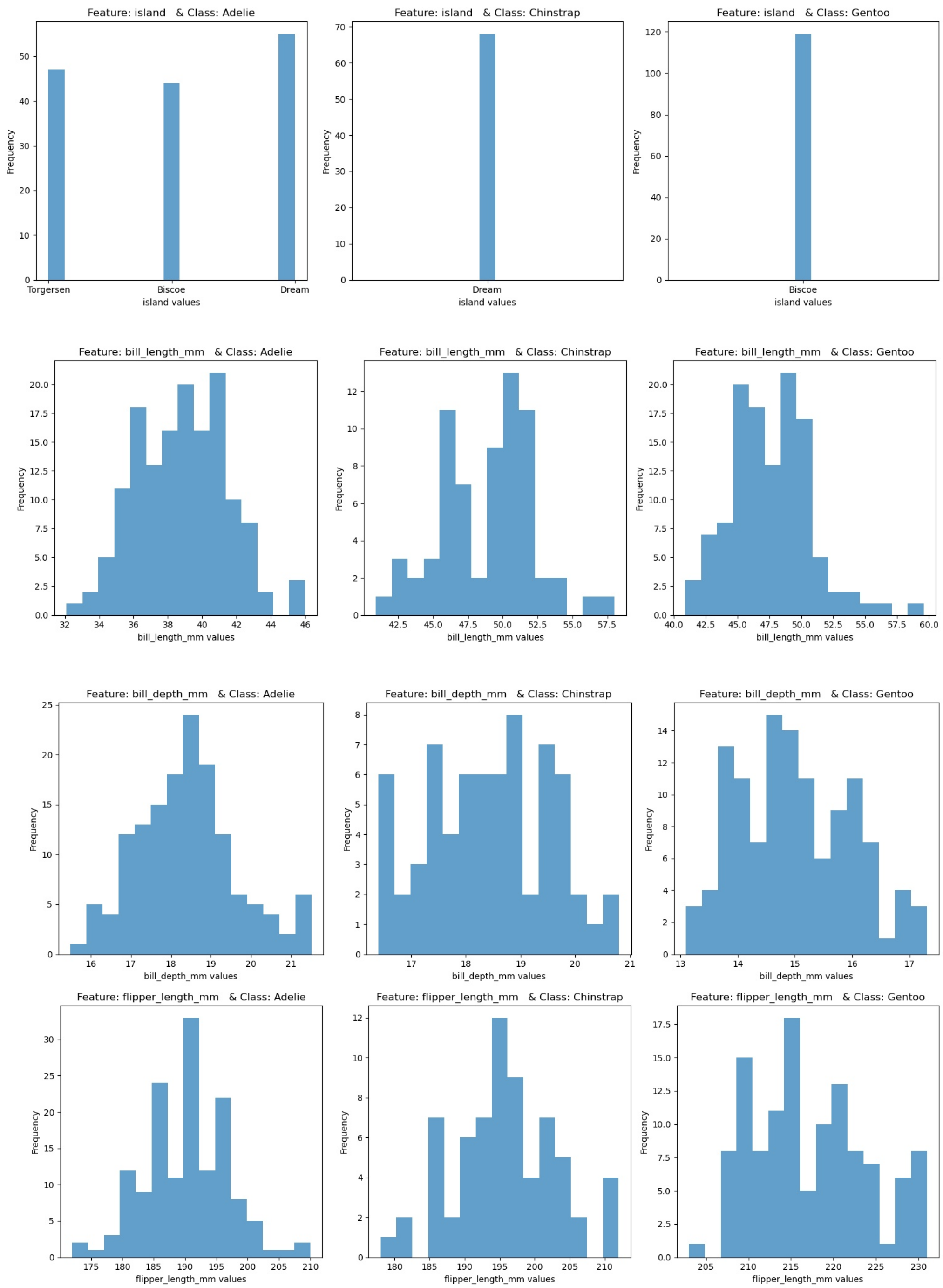
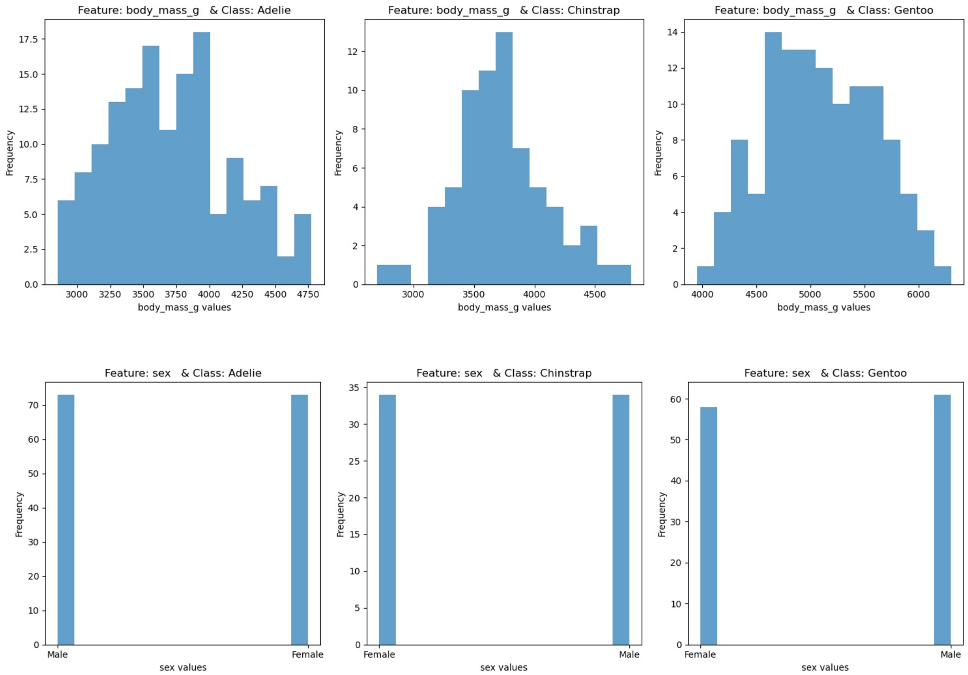
Notes: report for q2.1 would all be screenshots of my python script. I believe checking them in the script would be more convenient.

1.

```
Q2.2.1 Data Preprocessing
Number of data points in the whole dataset originally: 344
Number of data points in the whole dataset originally: 333
Number of data points in the training set after splitting: 233
Number of data points in the test set after splitting: 100
```

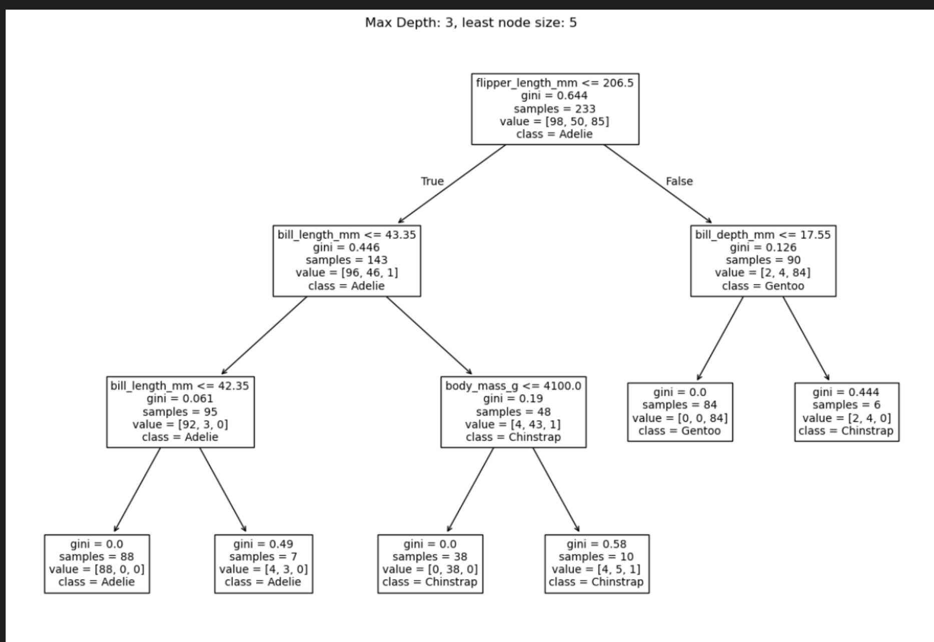
2.



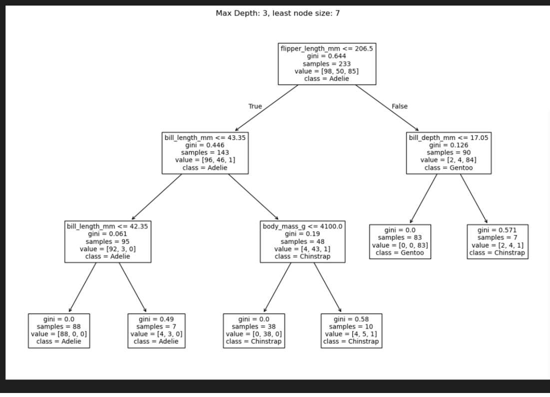


3.

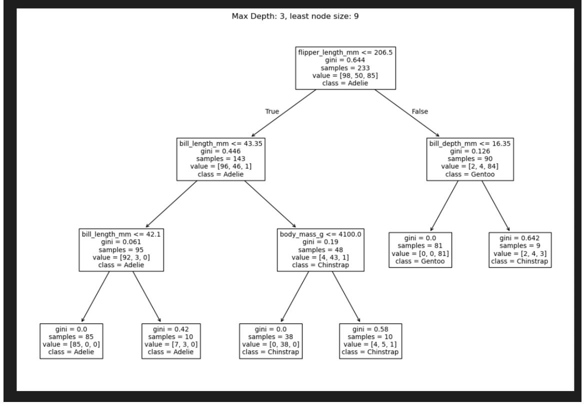
```
Q2.1.3 Decision Tree:
1. Results with maximum depth as 3 and least node size as 5:
training accuracy: 0.9571,  test accuracy: 0.9800
```



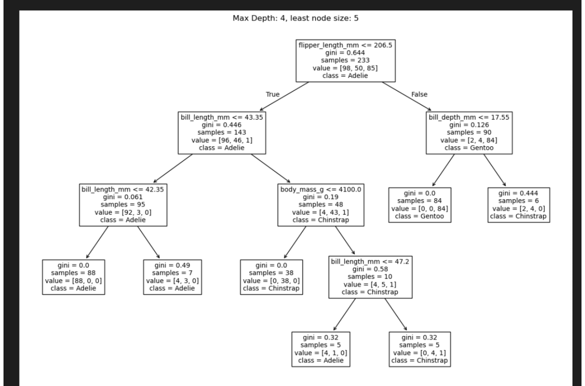
2. Results with maximum depth as 3 and least node size as 7:
training accuracy: 0.9528, test accuracy: 0.9600



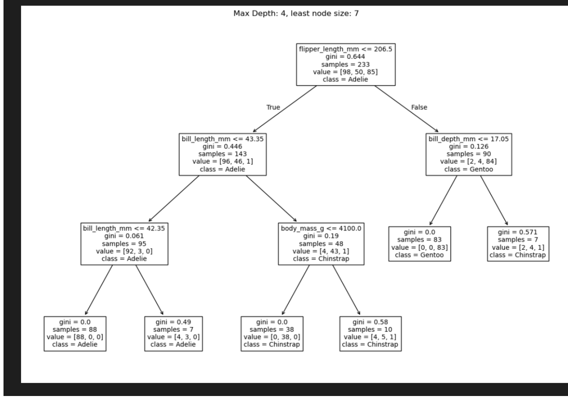
3. Results with maximum depth as 3 and least node size as 9:
training accuracy: 0.9442, test accuracy: 0.9200



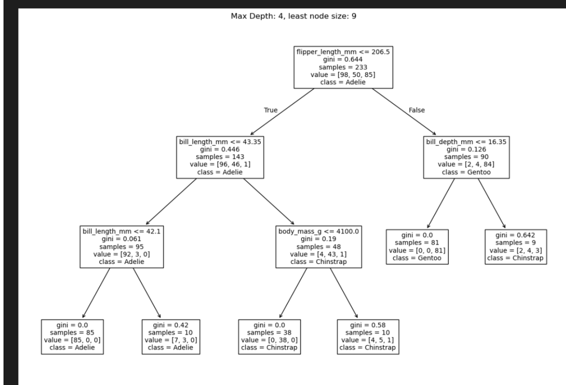
4. Results with maximum depth as 4 and least node size as 5:
training accuracy: 0.9700, test accuracy: 0.9800



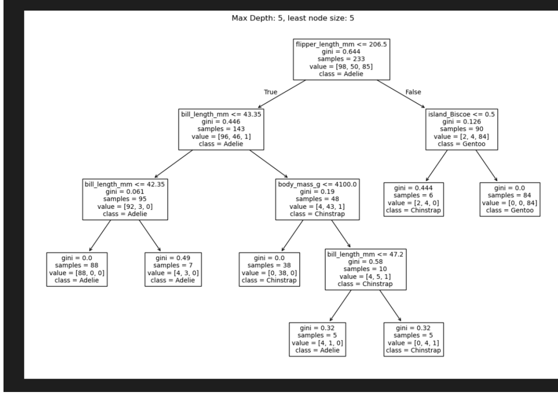
5. Results with maximum depth as 4 and least node size as 7:
training accuracy: 0.9528, test accuracy: 0.9600



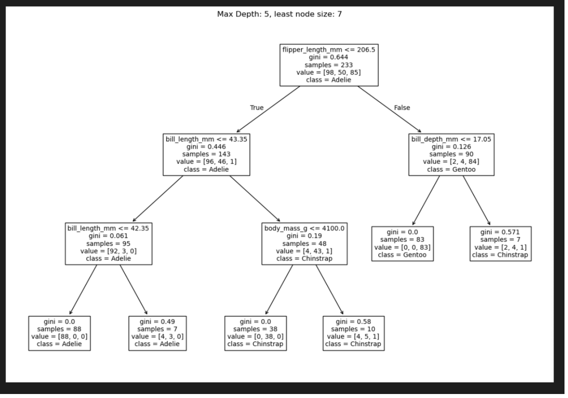
6. Results with maximum depth as 4 and least node size as 9:
training accuracy: 0.9442, test accuracy: 0.9200



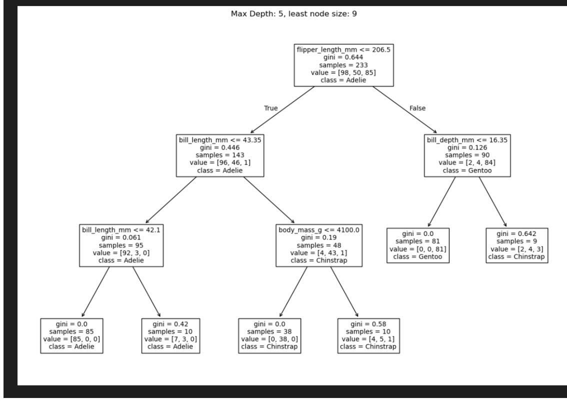
7. Results with maximum depth as 5 and least node size as 5:
training accuracy: 0.9700, test accuracy: 0.9800



```
8. Results with maximum depth as 5 and least node size as 7:
training accuracy: 0.9528,  test accuracy: 0.9600
```




```
9. Results with maximum depth as 5 and least node size as 9:
training accuracy: 0.9442,  test accuracy: 0.9200
```



4.

```
Q2.1.4 Bagging of Trees:
1. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 0.9871, test accuracy: 0.9800

2. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 0.9957, test accuracy: 0.9800

3. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 0.9957, test accuracy: 0.9800

4. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

5. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

6. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 0.9957, test accuracy: 0.9800

7. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

8. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

9. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900
```

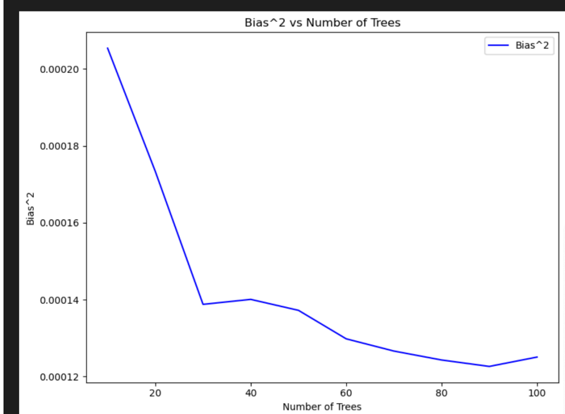
5.

```
Q2.1.5 Random Forests:
1. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

2. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9800

3. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

4. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

5. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

6. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9800

7. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

8. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9900

9. Results with maximum depth as 5 and number of trees as [50, 100, 150]:
training accuracy: 1.0000, test accuracy: 0.9800
```

6. This question may be divided into 2 scenarios, by the given example using tree = 10,20,...100. which is :

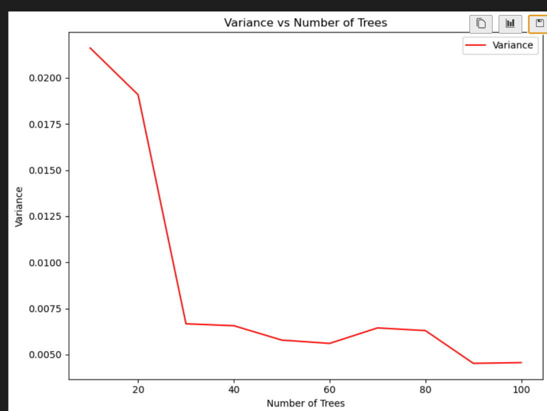
When the number of trees are relatively small, as the number of trees increases, the model predictions become more accurate, reducing bias.
When the number of trees are relatively small, as the number of trees increases, variance typically decreases. This is because the Random Forest model becomes more stable, averaging the predictions of multiple trees, which reduces the overall variance of the predictions.

```
Q2.1.6 Relationship of Bias/Variance with respect to Number of Trees in Random Forests:
Bias^2 - Trees:
```




```
When the number of trees are relatively small, as the number of trees increases,
the model predictions become more accurate, reducing bias.
```

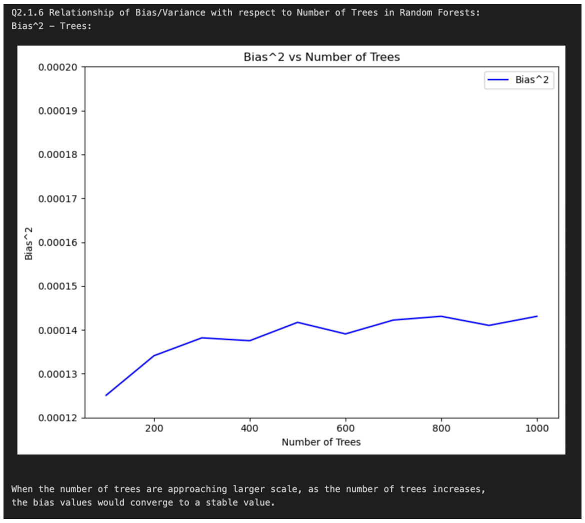
```
variance - trees
```

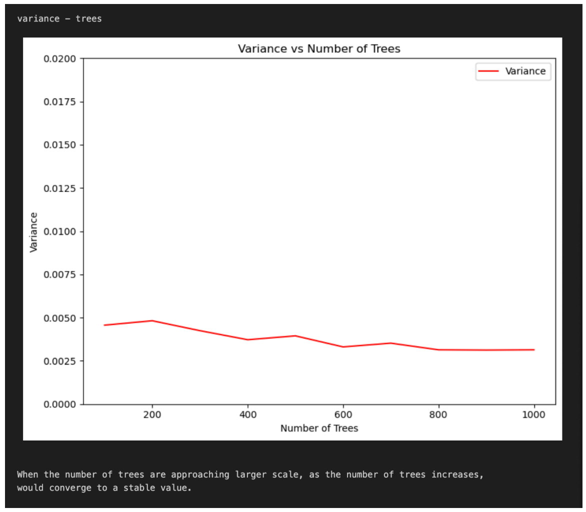



```
When the number of trees are relatively small, as the number of trees increases,
variance typically decreases. This is because the Random Forest model becomes more stable,
averaging the predictions of multiple trees, which reduces the overall variance of the predictions.
```

When the number of trees approach a larger scale, or extremely large, it would be:

When the number of trees are approaching larger scale, as the number of trees increases, the bias values would converge to a stable value.
When the number of trees are approaching larger scale, as the number of trees increases, variance would converge to a stable value.





Q2.2 The contents of this part would be the screenshot of the completed tables value & the conclusion of decision of the better model, as required. To see more details, please see attached python script for Q2.

Since the K fold process is shuffled, i.e. randomly split, the answer of mine displayed as:

3.

Logistic Regression							
	Hyper-paramter	1	2	3	4	5	
	penalty	l1	l1	l2	l2	l1	
Class 0	Metric	1	2	3	4	5	Avg
	Precision	0.928205	0.92268	0.875	0.89372	0.924623	0.90884569
	Recall	0.905	0.895	0.945	0.925	0.92	0.918
	F1	0.916456	0.908629	0.908654	0.909091	0.922306	0.91302713
Class1	Metric	1	2	3	4	5	Avg
	Precision	0.907317	0.898058	0.940217	0.92228	0.920398	0.9176541
	Recall	0.93	0.925	0.865	0.89	0.925	0.907
	F1	0.918519	0.91133	0.901042	0.905852	0.922693	0.91188718
Performace Evaluation	Metric	1	2	3	4	5	Avg
	Accuracy	0.9175	0.91	0.905	0.9075	0.9225	0.9125
	AUROC	0.947725	0.939175	0.962425	0.962025	0.956675	0.953605

4.

SVM							
	Hyper-paramter	1	2	3	4	5	
	C	1.00E-05	0.001	1.00E-05	0.0001	0.0001	
Class 0	Metric	1	2	3	4	5	Avg
	Precision	0.953125	0.928205	0.953846	0.958333	0.935	0.94570192
	Recall	0.915	0.905	0.93	0.92	0.935	0.921
	F1	0.933673	0.916456	0.941772	0.938776	0.935	0.93313537
Class1	Metric	1	2	3	4	5	Avg
	Precision	0.918269	0.907317	0.931707	0.923077	0.935	0.92307411
	Recall	0.955	0.93	0.955	0.96	0.935	0.947
	F1	0.936275	0.918519	0.94321	0.941176	0.935	0.93483588
Performace Evaluation	Metric	1	2	3	4	5	Avg
	Accuracy	0.935	0.9175	0.9425	0.94	0.935	0.934
	AUROC	0.9532	0.9444	0.96155	0.952975	0.9697	0.956365

5. As observed in average value comparison for both class0 & 1 and performance evaluation, we see the values of precision, recall, f1, accuracy, AUROC (all of them!!)of SVM model outperforms Logistic regression model for this dataset. The SVM's ability to effectively handle high-dimensional data and maximize the margin between classes likely contributes to its better performance.