Rutuja Dukhande
1001730748
CSE 6363: Machine Learning

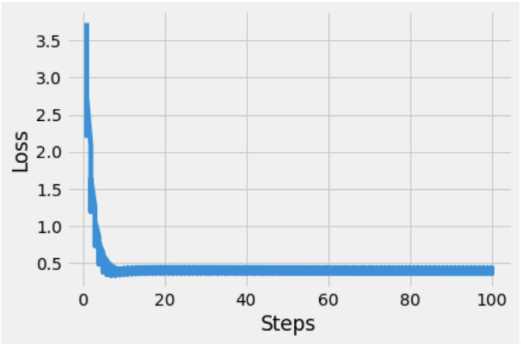
Assignment 1

Regression:

1. Model 1: input feature ‘sepal length’ to predict ‘sepal width’:

```
In [34]: #model1
clf = LinearRegression()
clf.fit(X_train[:, 0], X_train[:, 1])
predict = clf.predict(X_test[:, 0], X_test[:, 1])
print("The Test Loss:",predict)
```

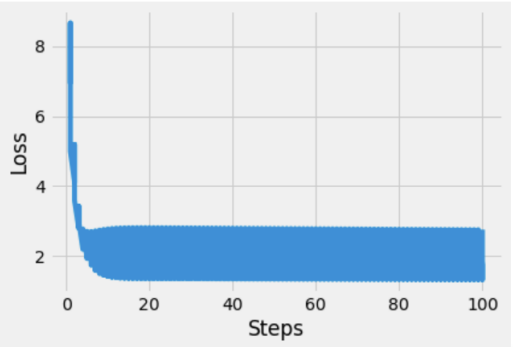
```
LOSS IS : 0.360691354936027
The Test Loss: 0.42742245355299663
```



2. Model 2: input feature ‘sepal length(cm)’ to predict ‘petal length(cm)’

```
In [35]: #model2
clf1 = LinearRegression()
clf1.fit(X_train[:, 0], X_train[:, 2])
predict1 = clf1.predict(X_test[:, 0], X_test[:, 2])
print("The Test Loss:",predict1)
```

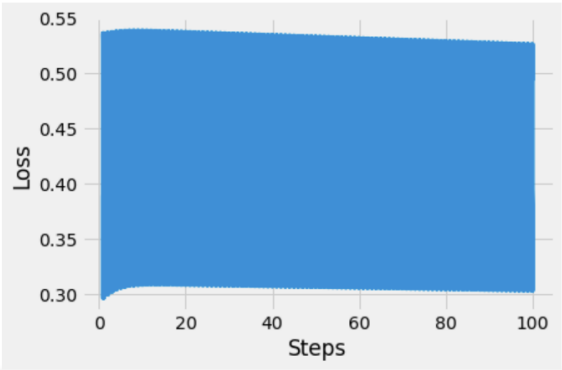
```
LOSS IS : 2.7499796644359376
The Test Loss: 1.5825071611326678
```



3. Model 3: input feature ‘sepal length(cm)’ to predict ‘petal width(cm)’.

```
In [36]: #model3
clf2 = LinearRegression()
clf2.fit(X_train[:, 0], X_train[:, 3])
predict2 = clf2.predict(X_test[:, 0], X_test[:, 3])
print("The Test Loss:",predict2)
```

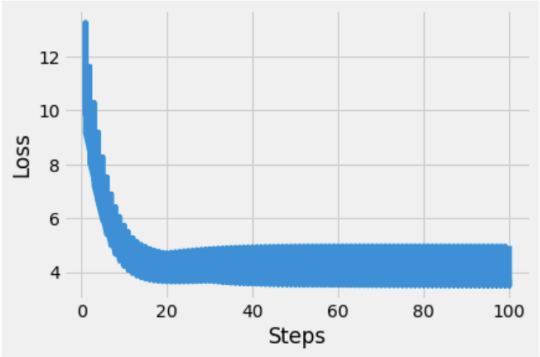
```
LOSS IS : 0.4953685329259942
The Test Loss: 0.3516786853122629
```



4. Model 4: input feature ‘sepal width(cm)’ to predict ‘petal length(cm)’

```
In [37]: #model4
clf3 = LinearRegression()
clf3.fit(X_train[:, 1], X_train[:, 2])
predict3 = clf3.predict(X_test[:, 1], X_test[:, 2])
print("The Test Loss:",predict3)
```

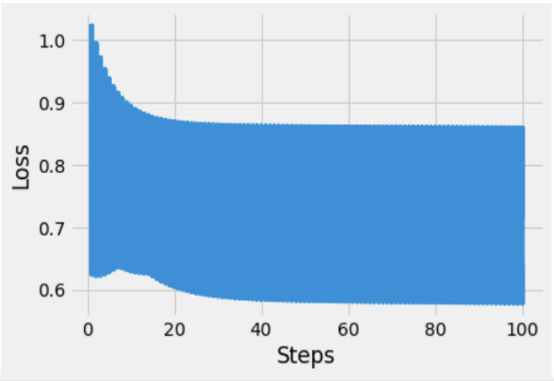
```
LOSS IS : 4.967119659498733
The Test Loss: 4.009234040912353
```



5. Model 5: input feature ‘sepal width(cm)’ to predict ‘petal width(cm)’

```
In [38]: #model5
clf4 = LinearRegression()
clf4.fit(X_train[:, 1], X_train[:, 3])
predict4 = clf4.predict(X_test[:, 1], X_test[:, 3])
print("The Test Loss:",predict4)
```

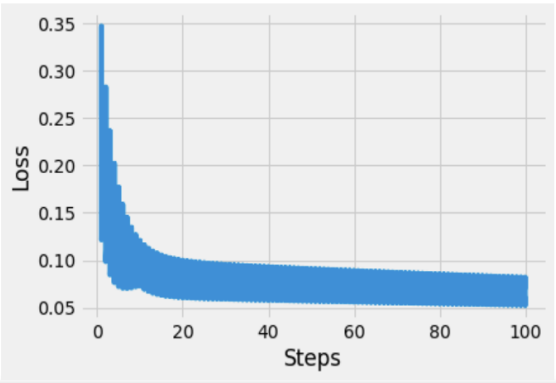
```
LOSS IS : 0.7155083037553959
The Test Loss: 0.5911023664389997
```



6. Model 6: input feature ‘petal length(cm)’ to predict ‘petal width(cm)’

```
In [39]: #model6
clf5 = LinearRegression()
clf5.fit(X_train[:, 2], X_train[:, 3])
predict5 = clf5.predict(X_test[:, 2], X_test[:, 3])
print("The Test Loss:",predict5)
```

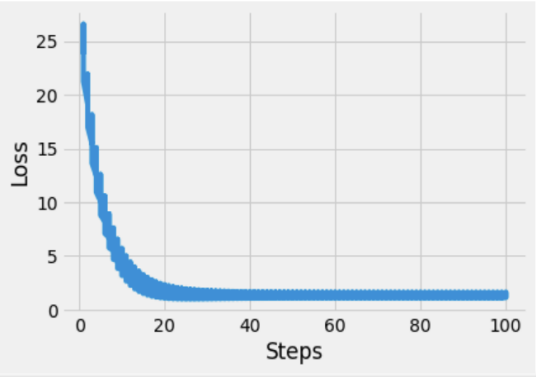
```
LOSS IS : 0.06975031870858704
The Test Loss: 0.10504993748525539
```



7. Model 7: input feature ‘sepal width(cm)’ to predict ‘sepal length(cm)’

```
In [40]: #model7
clf6 = LinearRegression()
clf6.fit(X_train[:, 1], X_train[:, 0])
predict6 = clf6.predict(X_test[:, 1], X_test[:, 0])
print("The Test Loss:",predict6)
```

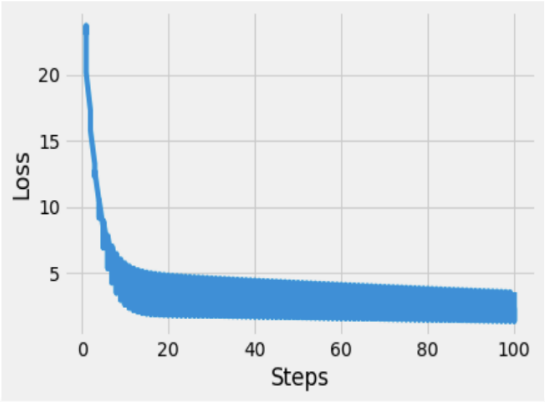
```
LOSS IS : 1.0981890228027442
The Test Loss: 1.557224212087829
```



8. Model 8: input feature ‘petal length(cm)’ to predict ‘sepal length(cm)’

```
In [41]: #model8
clf7 = LinearRegression()
clf7.fit(X_train[:, 2], X_train[:, 0])
predict7 = clf7.predict(X_test[:, 2], X_test[:, 0])
print("The Test Loss:",predict7)
```

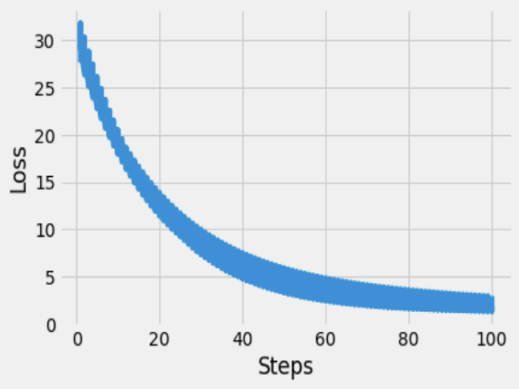
```
LOSS IS : 3.574977710483721
The Test Loss: 1.7534081822347347
```



9. Model 9: input feature ‘petal width(cm)’ to predict ‘sepal length(cm)’

```
In [42]: #model9
clf8 = LinearRegression()
clf8.fit(X_train[:, 3], X_train[:, 0])
predict8 = clf8.predict(X_test[:, 3], X_test[:, 0])
print("The Test Loss:",predict8)
```

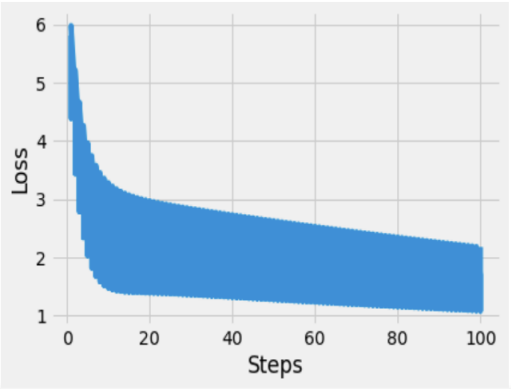
```
LOSS IS : 2.935775136997486
The Test Loss: 2.067874631263814
```



10. Model 10: input feature ‘petal length(cm)’ to predict ‘sepal width(cm)

```
In [43]: #model10
clf9 = LinearRegression()
clf9.fit(X_train[:, 2], X_train[:, 1])
predict9 = clf9.predict(X_test[:, 2], X_test[:, 1])
print("The Test Loss:",predict9)
```

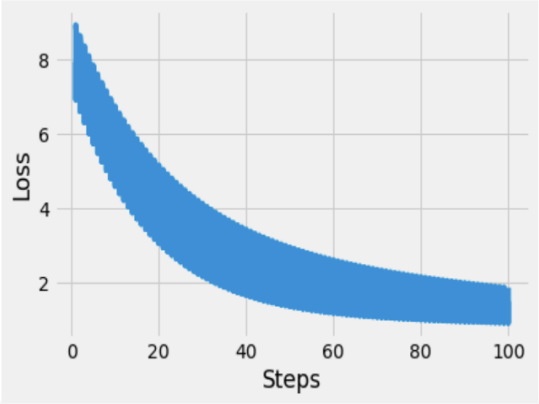
```
LOSS IS : 2.1732473467165128
The Test Loss: 1.2352392648028905
```



11. Model 11: input feature ‘petal width (cm)’ to predict ‘sepal width(cm)

```
In [44]: #model11
clf10 = LinearRegression()
clf10.fit(X_train[:, 3], X_train[:, 1])
predict10 = clf10.predict(X_test[:, 3], X_test[:, 1])
print("The Test Loss:",predict10)
```

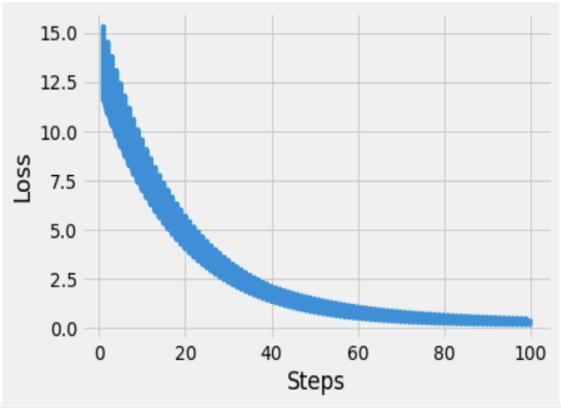
```
LOSS IS : 1.8725636712452547
The Test Loss: 1.02765598594766
```



12. Model 12: input feature ‘petal width (cm)’ to predict ‘petal length(cm)

```
In [45]: #model12
clf11 = LinearRegression()
clf11.fit(X_train[:, 3], X_train[:, 2])
predict11 = clf11.predict(X_test[:, 3], X_test[:, 2])
print("The Test Loss:",predict11)
```

```
LOSS IS : 0.46112599563517875
The Test Loss: 0.6499269585468069
```



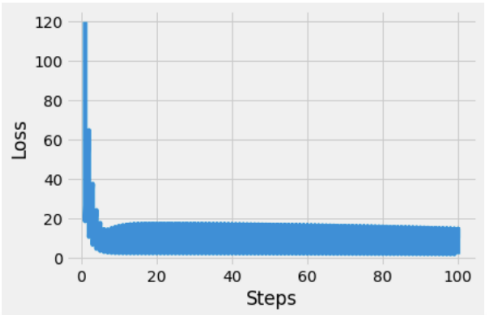
Models	Testing Accuracy
Model 1:'sepal length' to predict 'sepal width'	0.42742245355299663
Model 2: 'sepal length(cm)' to predict 'petal length(cm)'	1.5825071611326678
Model 3: 'sepal length(cm)' to predict 'petal width(cm)'	0.3516786853122629
Model 4: 'sepal width(cm)' to predict ' petal length(cm)'	4.009234040912353
Model 5: 'sepal width(cm)' to predict 'petal width(cm)'	0.5911023664389997
Model 6: 'petal length(cm)' to predict 'petal width(cm)'	0.10504993748525539
Model 7: 'sepal width(cm)' to predict ' sepal length(cm)'	1.557224212087829
Model 8: 'petal length(cm)' to predict 'sepal length(cm)'	1.7534081822347347
Model 9: 'petal width(cm)' to predict ' sepal length(cm)'	2.067874631263814
Model 10: 'petal length(cm)' to predict 'sepal width(cm)	1.2352392648028905
Model 11: 'petal width (cm)' to predict 'sepal width(cm)	1.02765598594766
Model 12: 'petal width (cm)' to predict 'petal length(cm)	0.6499269585468069

As per table above for testing accuracy, we can say model 6, i.e., Petal length input feature is a good predictive for corresponding petal width.

L2 Regularization:

In [65]:
```
#model12
clf12 = l2Regularization()
clf12.fit(X_train[:, 0], X_train[:, 1])
predict12 = clf12.predict(X_test[:, 0], X_test[:, 1])
print("The Test Loss:",predict12)
```

```
LOSS IS : 1.9471984059081078
The Test Loss: 0.48209196073950916
```



Classification:

Models	Testing Accuracy
Naive Bayes	1.0

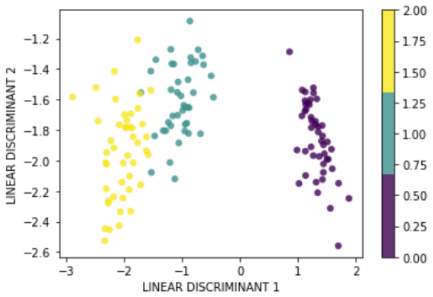
```
In [11]: nb = naiveBayes()
         nb.fit(X_train, y_train)
         y_prediction = nb.predict(X_test)
         #print(prediction)
         #nb.getAccuracy(y_test, prediction)

         def getAccuracy(y_new, y_p):
             return np.sum(y_new==y_p) / len(y_new)

         print("Accuracy is:",getAccuracy(y_test,y_prediction))

         Accuracy is: 1.0
```

Logistic Regression	0.8

```
In [56]: logiReg = LoggisticRegression(learningRate = 0.01, iterations=100 )
         logiReg.fit(X_train, y_train)
         y_prediction = logiReg.predict(X_test)

         def getAccuracy(y_new, y_p):
             return np.sum(y_new==y_p) / len(y_new)

         print("Accuracy is:",getAccuracy(y_test,y_prediction))

         Accuracy is: 0.8

In [ ]:
```

Linear Discriminant Analysis	1.0

```
SHAPE OF X: (150, 4)
AFTER TRANSFORMATION X: (135, 2)
```




```
In [11]: X_test_proj = lda.transform(X_test)
         model = LogisticRegression()
         model.fit(X_proj,y_train)

Out[11]: LogisticRegression()

In [12]: predictions = model.predict(X_test_proj)
         accuracy = np.sum(predictions == y_test)/len(y_test)
         print(accuracy)

         1.0
```

For classification, Naive Bayes and Linear Discriminant Analysis is the best classification model.