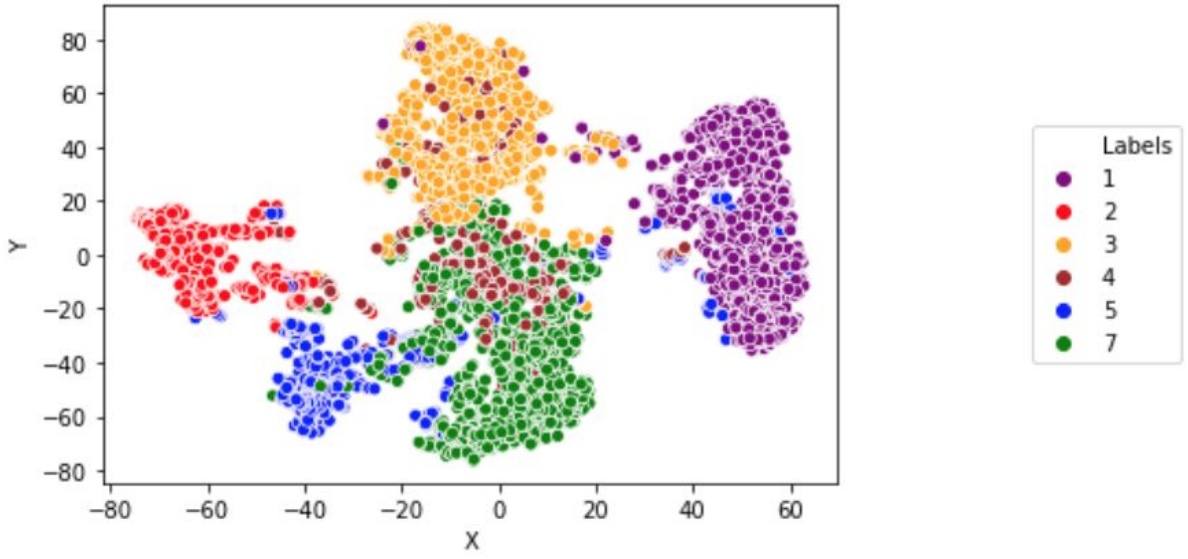
ML-CSE-543 – Machine Learning

Assignment 5 Analysis

By: Rajat Agarwal and Abhinav Saurabh Date: 25th Nov 2020

Q1

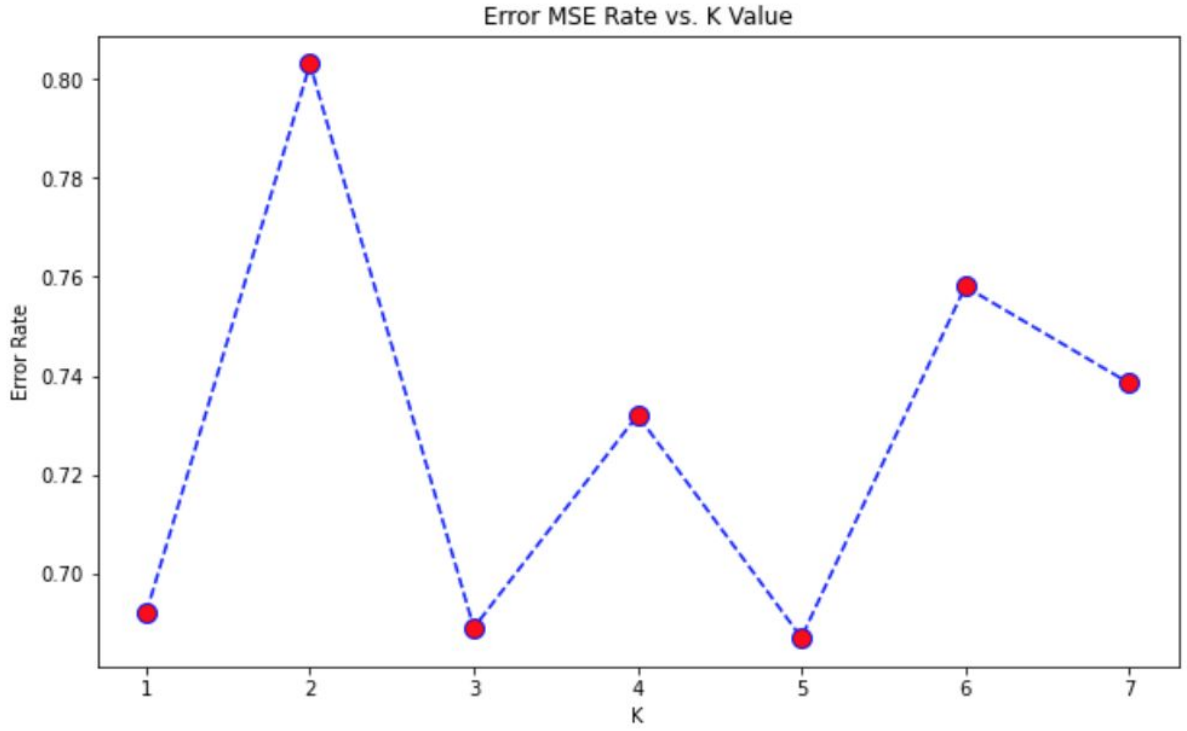
1. Dataset is already given in two parts training and testing data.
    - sat.trn is a training set which contains 4435 rows × 37 columns.
    - sat.tst is a test set which contains 2000 rows × 37 columns.
    - There are 36 features and 1 target variable.
    - We load the data using pandas library function read_csv.
    - Further divide the data into X_train,y_train and X_test,y_test
    - Then TSNE is applied to reduce the dimensionality to 2.

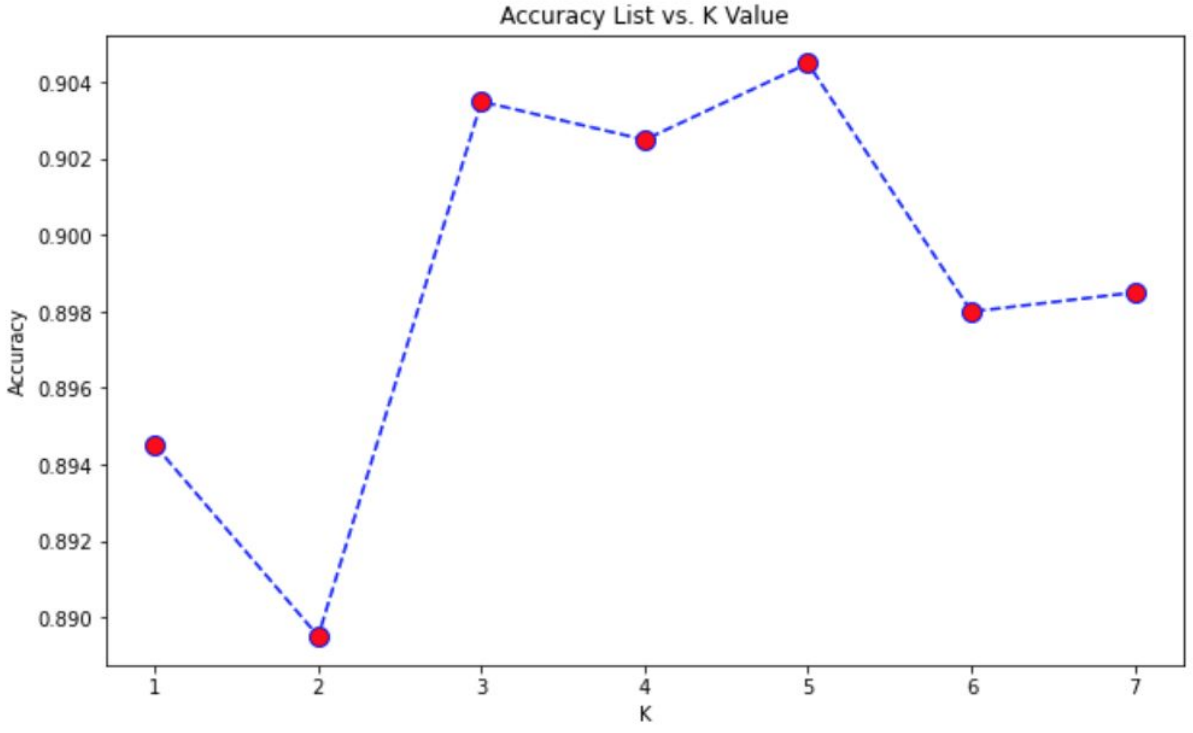


2. KNN is an instance based learning classification model.So here we kind of store the training examples first. When it runs into a new instance i.e during a test it tries to build relationships to stored training data to get target value for new instance.

    PseudoCode:
    1. Store training data first.
    2. Reiterate 3,4,5
    3. Locate K for the training set which is similar to test data.
    4. Put the most common class kNN into the y_pred.
    5. Go Step 2

    - We can find the best value of k using grid search.
    - For Errors we have used MSE calculate error.





From Above graph we can see that Accuracy is highest for k=5 , and Error rate is least for k=5.So by above we conclude that the optimal value for k is 5.
Highest Accuracy is received at K=5 i.e 90.45 on test data.
Lowest error mse is received at K=5 i.e 0.687.

3. Optimal value of K is 5. We obtained same accuracy on both sklearn accuracy and on scratch implementation.
    - Training Accuracy obtained was 94.11
    - Testing Accuracy obtained was 90.45

```
clf1  = KNN( K = 5)
clf1.fit(X_train.values, y_train.values)
```

```
trainpred = clf1.predict(X_train.values)
from sklearn.metrics import accuracy_scor
print('Accuracy:', accuracy_score(y_train
```

```
Accuracy: 0.9411499436302142
```

```
predictions = clf1.predict(X_test.values)
from sklearn.metrics import accuracy_scor
print('Accuracy:', accuracy_score(y_test,
```

```
Accuracy: 0.9045
```

```
from sklearn.neighbors import KNeighborsClass
clf = KNeighborsClassifier(n_neighbors = 5)
clf.fit(X_train, y_train)
```

```
<ipython-input-53-c485bdc695f0>:3: DataConver
f y to (n_samples, ), for example using ravel
  clf.fit(X_train, y_train)
KNeighborsClassifier()
```

```
trainpred = clf.predict(X_train)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_train, t
```

```
Accuracy: 0.9411499436302142
```

```
predictions = clf.predict(X_test)
from sklearn.metrics import accuracy_score
print('Accuracy:', accuracy_score(y_test, pr
```

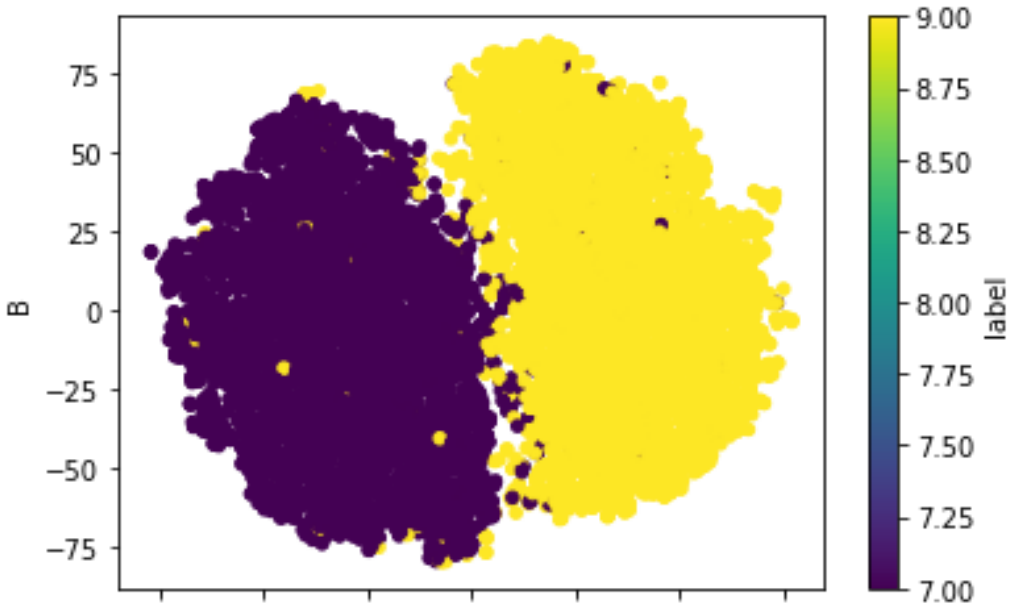
```
Accuracy: 0.9045
```

Q2

(1) Dataset consists of 14251 samples, 7293 belongs to class ‘7’ and 6958 belongs to class ‘9’. After 80:20 split 11400 and 2851 are divided into training and validation set respectively.

(2) Training and Testing accuracy are 0.9870 and 0.9845 respectively
Training and Testing loss (MSE) are 0.0491 and 0.07716 respectively

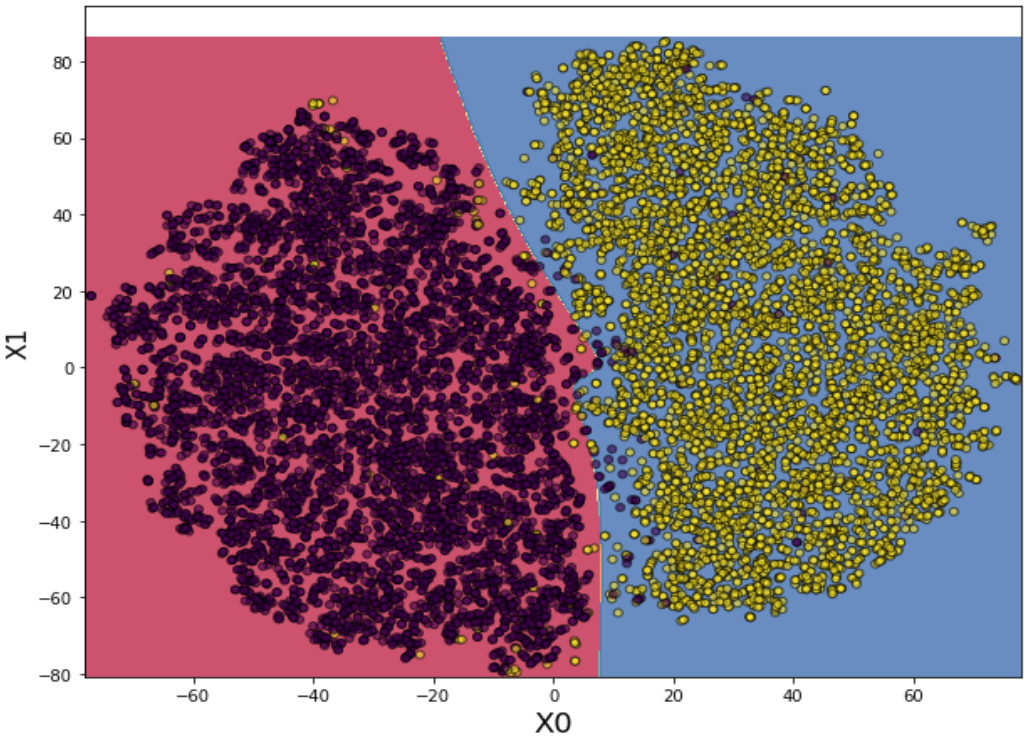
(3) The data is compressed to 2 features using TSNE, data is visualized as



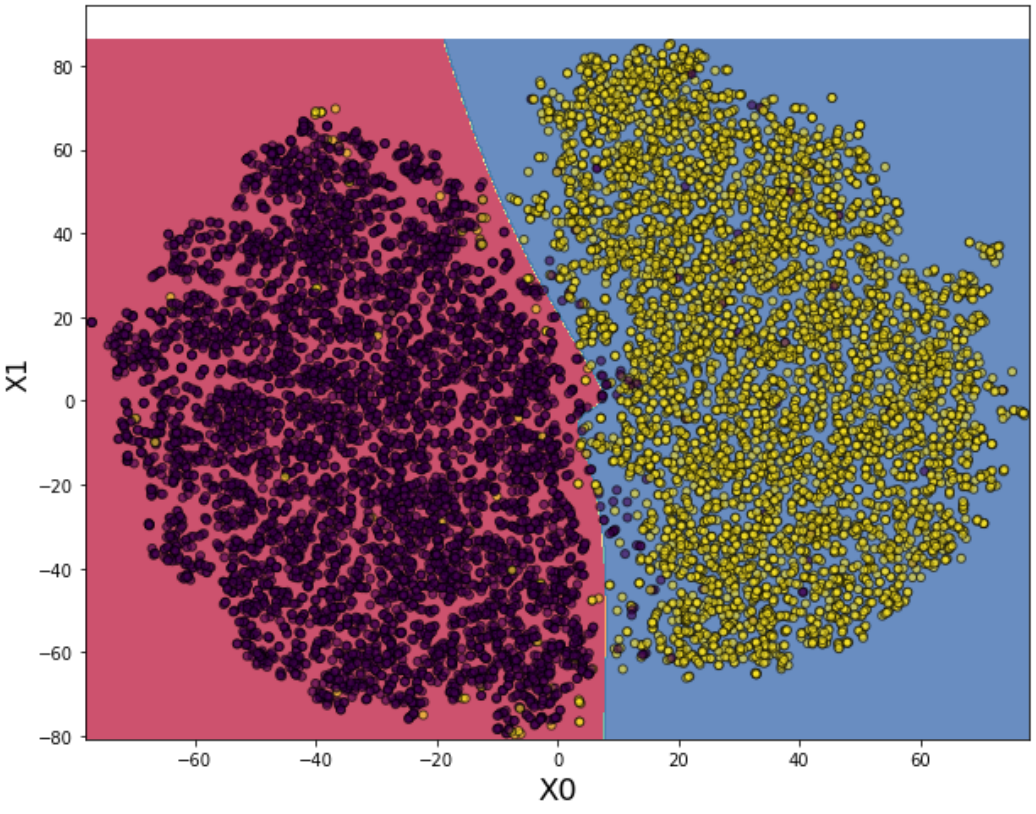
For different values of alpha (regularization value) following are the decision boundaries shown below. The plots show that different decision boundaries are formed for different values of alpha parameters. Increasing alpha leads to smaller weights and therefore the decision boundary is sharp (i.e. has less curvatures) and decreasing alpha values favours large weight and boundary formed has proper curvatures or less sharp.

We can generalize that higher alpha values are less complex the model is and it leads to increased bias and reduction in variance or fixes overfitting. Lower alpha values leads to more complex models and it may leads to overfitting the model therefore decision boundary is has high curvatures.

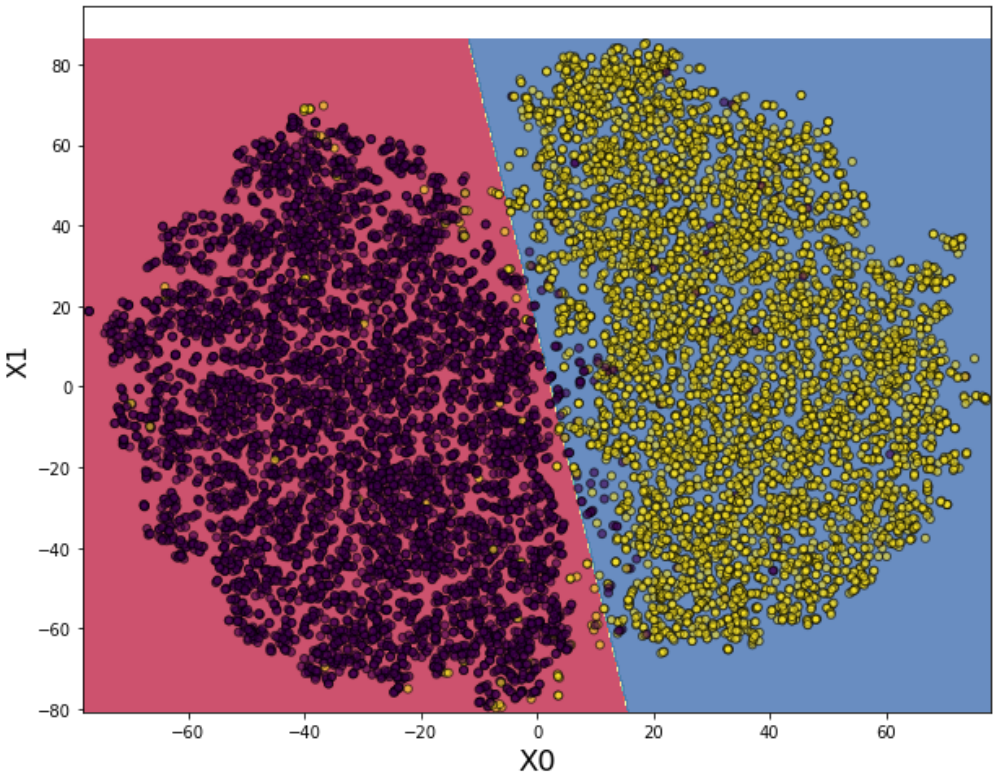
alpha 1e-07



alpha 0.0001



alpha 1



alpha 100