Final project

Shalymova Aigerim

Dataset

	Attribute1	Attribute2	Attribute3	Attribute4	Attribute5	Attribute6	١
0	80	102	102	79	76	102	
1	76	102	102	79	76	102	
2	80	98	106	79	76	94	
3	76	94	102	76	76	94	
4	76	94	102	76	76	94	
6430	56	64	108	96	64	71	
6431	64	71	108	96	68	75	
6432	68	75	108	96	71	87	
6433	71	87	108	88	71	91	
6434	71	91	100	81	76	95	
	Attribute7	Attribute8	Attribute9	Attribute10	Attri	bute28 \	
0	102	79	76	Attribute10 102	Attri	87	
1	102 106						
	102	79	76	102		87	
1	102 106	79 83	76 76	102 102		87 87	
1 2	102 106 102	79 83 76	76 76 76	102 102 94	•••	87 87 79	
1 2 3 4	102 106 102 102 102	79 83 76 76 76	76 76 76 76 76	102 102 94 94 89		87 87 79 79 75	
1 2 3	102 106 102 102 102	79 83 76 76 76	76 76 76 76 76	102 102 94 94 89 		87 87 79 79 75	
1 2 3 4 6430 6431	102 106 102 102 102	79 83 76 76 76	76 76 76 76 76 68 71	102 102 94 94 89		87 87 79 79 75	
1 2 3 4 6430	102 106 102 102 102 	79 83 76 76 76 96	76 76 76 76 76	102 102 94 94 89 		87 87 79 79 75	
1 2 3 4 6430 6431	102 106 102 102 102 108 108	79 83 76 76 76 96 96	76 76 76 76 76 68 71	102 102 94 94 89 75		87 87 79 79 75 92 96	
1 2 3 4 6430 6431 6432	102 106 102 102 102 108 108	79 83 76 76 76 96 96	76 76 76 76 76 68 71 71	102 102 94 94 89 75 87 91		87 87 79 79 75 92 96 89	

Dataset contains of 6435 rows and 37 columns. There are 36 features and 6 classes.

Data

```
import pandas as pd
import numpy as np

df = pd.read_csv('/Users/asik/Downloads/satellite.csv')
# check NA values
df.isnull().sum()
```

There were no empty values

: Attribute1 Attribute2 0 Attribute3 Attribute4 Attribute5 Attribute6 Attribute7 Attribute8 Attribute9 Attribute10 Attribute11 Attribute12 Attribute13 Attribute14 Attribute15 Attribute16 Attribute17 Attribute18 Attribute19 Attribute20 Attribute21 Attribute22 Attribute23 Attribute24 Attribute25 Attribute26 Attribute27 Attribute28 Attribute29 Attribute30 Attribute31 Attribute32 0

Neural Network

```
0.40
X, y = df.drop("Class", axis = 1), df["Class"]
                                                                                                                                                                                           Training error
X train, X test, y train, y test = train test split(X, y, test size=0.1, random state=1)
                                                                                                                      0.35
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
                                                                                                                   accuracy)
                                                                                                                      0.30
                                                                                                  □ ↑
#NN without normalization
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score
NN_model = MLPClassifier(random_state=0, max_iter=1000)
                                                                                                                      0.25
NN model.fit(X train.v train)
                                                                                                                   Error (1
y_pred1=NN_model.predict(X_valid)
                                               train_error=[]
                                                                                                                      0.20
y_pred2=NN_model.predict(X_test)
                                                 valid error=[]
                                                 m=[1,1000,2000,3000,5000,6000]
accuracy1=accuracy_score(y_pred1,y_valid)
                                                 for j in m:
accuracy2=accuracy_score(y_pred2,y_test)
                                                     NN_model = MLPClassifier(random_state=42, max_iter=1000)
                                                                                                                      0.15
print('Accuracy of validation set:',accuracy1)
                                                     NN_model.fit(X_train[:j],y_train[:j])
print('Accuracy of test set:',accuracy2)
                                                     y_pred1=NN_model.predict(X_valid)
                                                    y_pred2=NN_model.predict(X_train[:j])
Accuracy of validation set: 0.8275862068965517
                                                                                                                      0.10
                                                     accuracy1=accuracy score(y pred1,y valid)
Accuracy of test set: 0.8090062111801242
                                                     accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                     train error.append(1-accuracy2)
                                                                                                                                          1000
                                                                                                                                                      2000
                                                                                                                                                                   3000
                                                                                                                                                                                                        6000
                                                                                                                                                                               4000
                                                                                                                                                                                            5000
                                                     valid_error.append(1-accuracy1)
                                                                                                                                                            Training Set Size
                                              : import matplotlib.pyplot as plt
                                                 from scipy.ndimage import gaussian_filter1d
                                                 plt.plot(m,gaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                 plt.plot(m, gaussian filter1d(train error, sigma=2), label='Training error')
                                                 plt.legend()
                                                 plt.xlabel('Training Set Size')
                                                 plt.ylabel('Error (1 - accuracy)')
```

Validation error

Neural Network

```
Validation error
                                                                                                                       0.40
X, y = df.drop("Class", axis = 1), df["Class"]
                                                                                                                                                                                         Training error
 \#X, y = df.drop(["quality","Id"], axis = 1), <math>df["quality"]
 #normalization
 for column in X.columns:
                                                                                                                      0.35
     X[column] = X[column] / X[column].abs().max()
 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                                   accuracy)
                                                                                                                       0.30
 X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
                                                                                                                       0.25
 # NN with normalization and dif parameters
 from sklearn.neural_network import MLPClassifier
                                                                                                                   Error (1
 from sklearn.metrics import accuracy_score
                                                                                                                       0.20
 NN model = MLPClassifier(hidden layer sizes=(57,), random state=0, max iter=2000)
 NN_model.fit(X_train,y_train)
                                                   train error=[]
                                                   valid error=[]
 y_pred1=NN_model.predict(X_valid)
                                                   m = [1, 1000, 2000, 3000, 5000, 6000]
 y_pred2=NN_model.predict(X_test)
                                                       NN_model = MLPClassifier(hidden_layer_sizes=(57,),random_state=0, max_iter=1000)
 accuracy1=accuracy_score(y_pred1,y_valid)
                                                       NN_model.fit(X_train[:j],y_train[:j])
 accuracy2=accuracy_score(y_pred2,y_test)
                                                       y_pred1=NN_model.predict(X_valid)
 print('Accuracy of validation set:',accuracy1)
                                                       y_pred2=NN_model.predict(X_train[:j])
 print('Accuracy of test set:',accuracy2)
                                                       accuracy1=accuracy_score(y_pred1,y_valid)
                                                                                                                                          1000
                                                                                                                                                     2000
                                                                                                                                                                 3000
                                                                                                                                                                              4000
                                                                                                                                                                                          5000
                                                                                                                                                                                                     6000
                                                       accuracy2=accuracy_score(y_pred2,y_train[:j])
 Accuracy of validation set: 0.8981191222570533
                                                                                                                                                           Training Set Size
 Accuracy of test set: 0.906832298136646
                                                       train_error.append(1-accuracy2)
                                                       valid error.append(1-accuracy1)
                                                                                                                                      Here I added normalization and changed
                                                   import matplotlib.pyplot as plt
                                                                                                                                      the parameter hidden layer to 57. It
                                                   from scipy.ndimage import gaussian_filter1d
                                                   plt.plot(m,gaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                                                                                                      helped me increase accuracy from 0.82
                                                   plt.plot(m,qaussian filter1d(train error,sigma=2),label='Training error')
                                                   plt.legend()
                                                   plt.xlabel('Training Set Size')
                                                                                                                                      to 0.90
                                                   plt.ylabel('Error (1 - accuracy)')
```

Decision tree

```
Validation error
df = pd.read_csv('/Users/asik/Downloads/satellite.csv')
                                                                                                                                                                                    Training error
df['Class'] = df['Class'].replace({7 : 6})
                                                                                                                      0.20
X, y = df.drop("Class", axis = 1), df["Class"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                                   accuracy)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
                                                                                                                      0.15
from sklearn.tree import DecisionTreeClassifier
# Modeling
                                                                                                                      0.10
clf = DecisionTreeClassifier(random_state=0)
                                                                                                                   Error
clf.fit(X_train,y_train)
                                                ]: train error=[]
y_pred1 = clf.predict(X_valid)
                                                    valid error=[]
y pred2 = clf.predict(X test)
                                                    m = [100, 1000, 2000, 3000, 5000, 6000]
accuracy1=accuracy score(y pred1,y valid)
                                                    for j in m:
                                                                                                                      0.05
accuracy2=accuracy_score(y_pred2,y_test)
                                                        clf = DecisionTreeClassifier(random state=0)
print('Accuracy of validation set:',accuracy1)
                                                        clf.fit(X train[:j], y train[:j])
print('Accuracy of test set:',accuracy2)
                                                        y_pred1=clf.predict(X_valid)
Accuracy of validation set: 0.8824451410658307
                                                                                                                      0.00
                                                        y_pred2=clf.predict(X_train[:j])
Accuracy of test set: 0.8788819875776398
                                                                                                                                                              3000
                                                                                                                                       1000
                                                                                                                                                  2000
                                                                                                                                                                         4000
                                                                                                                                                                                    5000
                                                                                                                                                                                                6000
                                                        accuracy1=accuracy_score(y_pred1,y_valid)
                                                        accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                                                                                                                        Training Set Size
                                                        train_error_append(1-accuracy2)
                                                        valid_error.append(1-accuracy1)
                                                ]: import matplotlib.pyplot as plt
                                                    from scipy.ndimage import gaussian_filter1d
                                                    plt.plot(m,qaussian_filter1d(valid_error, sigma=1),label='Validation error')
                                                    plt.plot(m,qaussian_filter1d(train_error,sigma=1),label='Training error')
                                                    plt.legend()
                                                    plt.xlabel('Training Set Size')
                                                    plt.ylabel('Error (1 - accuracy)')
```

Decision tree

```
0.175
X, y = df.drop("Class", axis = 1), df["Class"]
#normalization
                                                                                                               0.150
for column in X.columns:
   X[column] = X[column] / X[column].abs().max()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                               0.125
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
#DEcision tree with norm
                                                                                                               0.100
from sklearn.tree import DecisionTreeClassifier
                                                                                                            Error
clf = DecisionTreeClassifier(random_state=0, max_depth=9)
                                                                                                               0.075
clf.fit(X train,y train)
y_pred1 = clf.predict(X_valid)
                                              : train_error=[]
                                                valid_error=[]
y pred2 = clf.predict(X test)
                                                m = [100, 1000, 2000, 3000, 5000, 6000]
                                                for j in m:
accuracy1=accuracy_score(y_pred1,y_valid)
                                                    clf = DecisionTreeClassifier(random_state=0, max_depth=9)
accuracy2=accuracy score(y pred2,y test)
                                                    clf.fit(X_train[:j],y_train[:j])
print('Accuracy of validation set:',accuracy1)
                                                    y pred1=clf.predict(X valid)
print('Accuracy of test set:',accuracy2)
                                                    y pred2=clf.predict(X train[:j])
                                                    accuracy1=accuracy_score(y_pred1,y_valid)
                                                                                                                                    1000
                                                                                                                                                             3000
                                                                                                                                                2000
                                                                                                                                                                          4000
Accuracy of validation set: 0.8996865203761756
                                                    accuracy2=accuracy_score(y_pred2,y_train[:j])
Accuracy of test set: 0.8726708074534162
                                                                                                                                                       Training Set Size
                                                    train_error.append(1-accuracy2)
                                                    valid_error.append(1-accuracy1)
                                                                                                                         Here I added normalization and changed
                                                import matplotlib.pyplot as plt
                                                                                                                         parameter max_depth to 9 and it helped to
                                                from scipy.ndimage import gaussian_filter1d
                                                plt.plot(m,gaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                                                                                        improve learning curve and increased
                                                plt.plot(m,gaussian_filter1d(train_error,sigma=2),label='Training error')
                                                plt.legend()
                                                plt.xlabel('Training Set Size')
                                                                                                                         accuracy from 0.88 to 0.90.
                                                plt.ylabel('Error (1 - accuracy)')
```

Validation error Training error

6000

5000

Random Forest

plt.xlabel('Training Set Size')
plt.ylabel('Error (1 - accuracy)')

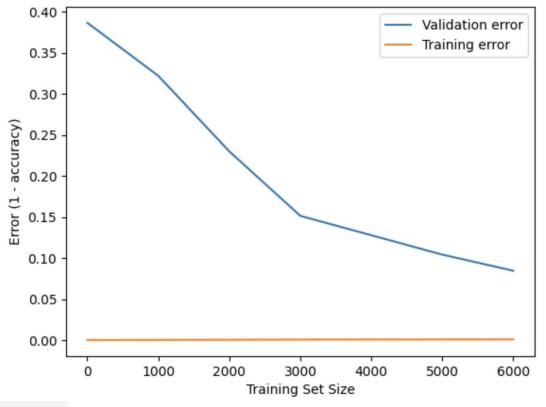
```
0.16
                                                                                                                                                                                     Validation error
df = pd.read_csv('/Users/asik/Downloads/satellite.csv')
                                                                                                                                                                                     Training error
                                                                                                                     0.14
df['Class'] = df['Class'].replace({7 : 6})
X, y = df.drop("Class", axis = 1), df["Class"]
                                                                                                                      0.12
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                                  accuracy)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
                                                                                                                     0.10
#Random Forest without normalization
from sklearn.ensemble import RandomForestClassifier
                                                                                                                      0.08
clf = RandomForestClassifier(max_depth=15, random_state=0)
                                                                                                                     0.06
clf.fit(X train, y train)
                                                train error=[]
y_pred1 = clf.predict(X_valid)
                                                valid error=[]
                                                                                                                      0.04
y pred2 = clf.predict(X test)
                                                m = [100, 1000, 2000, 3000, 5000, 6000]
                                                for j in m:
accuracy1=accuracy_score(y_pred1,y_valid)
                                                    clf = RandomForestClassifier(max depth=15, random state=0)
                                                                                                                      0.02
accuracy2=accuracy score(y pred2,y test)
                                                    clf.fit(X_train[:j],y_train[:j])
print('Accuracy of validation set:',accuracy1)
                                                    y_pred1=clf.predict(X_valid)
print('Accuracy of test set:',accuracy2)
                                                                                                                      0.00
                                                    y pred2=clf.predict(X train[:j])
Accuracy of validation set: 0.932601880877743
                                                    accuracy1=accuracy_score(y_pred1,y_valid)
Accuracy of test set: 0.9332298136645962
                                                                                                                                       1000
                                                                                                                                                  2000
                                                                                                                                                              3000
                                                                                                                                                                          4000
                                                                                                                                                                                      5000
                                                                                                                                                                                                  6000
                                                    accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                                                                                                                         Training Set Size
                                                    train error.append(1-accuracy2)
                                                    valid_error.append(1-accuracy1)
                                                import matplotlib.pyplot as plt
                                                from scipy.ndimage import gaussian_filter1d
                                                plt.plot(m,gaussian_filter1d(valid_error, sigma=1),label='Validation error')
                                                plt.plot(m,gaussian_filter1d(train_error,sigma=1),label='Training error')
                                                plt.legend()
```

Random Forest

```
X, y = df.drop("Class", axis = 1), df["Class"]
#normalization
for column in X.columns:
    X[column] = X[column] / X[column].abs().max()
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                                      accuracy)
X_train, X_valid, y_train, y_valid = train_test_split(X_train, y_train, test_size=0.11, random_state=1)
#Random Forest with normalization
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(max_depth=15, random_state=0)
clf.fit(X_train, y_train)
                                                  train_error=[]
y_pred1 = clf.predict(X_valid)
                                                  valid error=[]
y_pred2 = clf.predict(X_test)
                                                  m=[1,1000,2000,3000,5000,6000]
                                                  for j in m:
accuracy1=accuracy_score(y_pred1,y_valid)
                                                      clf = RandomForestClassifier(max_depth=15, random_state=0)
accuracy2=accuracy score(y pred2,y test)
                                                      clf.fit(X_train[:j],y_train[:j])
print('Accuracy of validation set:',accuracy1)
print('Accuracy of test set:',accuracy2)
                                                      y_pred1=clf.predict(X_valid)
                                                      y_pred2=clf.predict(X_train[:j])
Accuracy of validation set: 0.9294670846394985
                                                      accuracy1=accuracy_score(y_pred1,y_valid)
Accuracy of test set: 0.9332298136645962
                                                      accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                      train_error.append(1-accuracy2)
                                                      valid_error.append(1-accuracy1)
                                                  import matplotlib.pyplot as plt
                                                  from scipy.ndimage import gaussian_filter1d
                                                  plt.plot(m,gaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                  plt.plot(m,gaussian_filter1d(train_error,sigma=2),label='Training error')
```

plt.legend()

plt.xlabel('Training Set Size')
plt.ylabel('Error (1 - accuracy)')



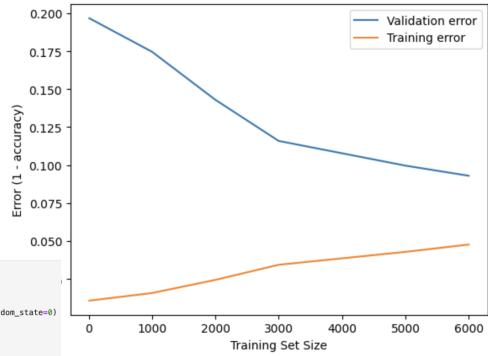
Here I just added normalization and it improved the learning curve.

Gradient Boosting

```
Validation error
df = pd.read_csv('/Users/asik/Downloads/satellite.csv')
                                                                                                                                                                                                       Training error
df['Class'] = df['Class'].replace({7 : 6})
                                                                                                                                        0.70
X, y = df.drop("Class", axis = 1), df["Class"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
                                                                                                                                        0.65
                                                                                                                                    accuracy)
X train, X valid, y train, y valid = train test split(X train, y train, test size=0.11, random state=1)
from sklearn.ensemble import GradientBoostingClassifier
                                                                                                                                       0.60
clf = GradientBoostingClassifier(n_estimators=100, learning_rate=0.001,max_depth=1, random_state=0)
                                                                                                                                    Error (1
clf.fit(X train, v train)
y_pred1 = clf.predict(X_valid)
y_pred2 = clf.predict(X_test)
accuracy1=accuracy score(y pred1,y valid)
accuracy2=accuracy_score(y_pred2,y_test)
                                                                                                                                        0.50
                                                           train_error=[]
print('Accuracy of validation set:',accuracy1)
                                                           valid_error=[]
print('Accuracy of test set:',accuracy2)
                                                           m = [10, 1000, 2000, 3000, 5000, 6000]
                                                           for j in m:
Accuracy of validation set: 0.6332288401253918
                                                              clf = GradientBoostingClassifier(n_estimators=50, learning_rate=0.001,max_depth=1
                                                                                                                                       0.45
                                                              clf.fit(X_train[:j],y_train[:j])
Accuracy of test set: 0.6304347826086957
                                                              y_pred1=clf.predict(X_valid)
                                                                                                                                                                     2000
                                                                                                                                                                                3000
                                                                                                                                                          1000
                                                                                                                                                                                            4000
                                                                                                                                                                                                        5000
                                                                                                                                                                                                                   6000
                                                              y pred2=clf.predict(X train[:j])
                                                                                                                                                                           Training Set Size
                                                              accuracy1=accuracy_score(y_pred1,y_valid)
                                                              accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                              train_error.append(1-accuracy2)
                                                              valid_error.append(1-accuracy1)
                                                                                                                                           □ 个、
                                                           import matplotlib.pyplot as plt
                                                           from scipy.ndimage import gaussian_filter1d
                                                          plt.plot(m,qaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                          plt.plot(m,gaussian_filter1d(train_error,sigma=2),label='Training error')
                                                           plt.legend()
                                                           plt.xlabel('Training Set Size')
                                                          plt.ylabel('Error (1 - accuracy)')
```

Gradient Boosting

```
0.200
X, y = df.drop("Class", axis = 1), df["Class"]
 #normalization
                                                                                                                                         0.175
 for column in X.columns:
     X[column] = X[column] / X[column].abs().max()
                                                                                                                                         0.150
 X train, X test, y train, y test = train test split(X, y, test size=0.1, random state=1)
                                                                                                                                         0.125
 X train, X valid, y train, y valid = train test split(X train, y train, test size=0.11, random state=1)
 from sklearn.ensemble import GradientBoostingClassifier
                                                                                                                                         0.100
 clf = GradientBoostingClassifier(n_estimators=1000, learning_rate=0.1,max_depth=1, random_state=0)
 clf.fit(X_train, y_train)
                                                                                                                                         0.075
 y_pred1 = clf.predict(X_valid)
 y_pred2 = clf.predict(X_test)
                                                                                                                                         0.050
accuracy1=accuracy_score(y_pred1,y_valid)
accuracy2=accuracy_score(y_pred2,y_test)
                                                          train_error=[]
print('Accuracy of validation set:',accuracy1)
                                                          valid_error=[]
                                                         m = [10, 1000, 2000, 3000, 5000, 6000]
print('Accuracy of test set:',accuracy2)
                                                             clf=GradientBoostingClassifier(n_estimators=1000, learning_rate=0.1,max_depth=1, random_state=0)
 Accuracy of validation set: 0.9106583072100314
                                                             clf.fit(X_train[:j],y_train[:j])
 Accuracy of test set: 0.9083850931677019
                                                             y_pred1=clf.predict(X_valid)
                                                             y_pred2=clf.predict(X_train[:j])
                                                             accuracy1=accuracy_score(y_pred1,y_valid)
                                                             accuracy2=accuracy_score(y_pred2,y_train[:j])
                                                             train_error.append(1-accuracy2)
                                                             valid_error.append(1-accuracy1)
                                                       : import matplotlib.pyplot as plt
                                                         from scipy.ndimage import gaussian_filter1d
                                                         plt.plot(m,gaussian_filter1d(valid_error, sigma=2),label='Validation error')
                                                         plt.plot(m,gaussian_filter1d(train_error,sigma=2),label='Training error')
                                                         plt.legend()
                                                         plt.xlabel('Training Set Size')
                                                         plt.ylabel('Error (1 - accuracy)')
```



I added normalization and changed parameters like learning rate and number of estimators so it increased accuracy from 0.63 to 0.91 and improved learning curve respectively.

Comparison table

Methods	Accuracy without normalization	Accuracy with normalization + different parameters changed
Neural network validation	0.827	0.898
test	0.809	0.906
Decision tree validation	0.882	0.90
test	0.879	0.872
Random Forest validation	0.933	0.929
test	0.933	0.933
Gradient Boosting validation	0.633	0.91
test	0.630	0.908