```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
import pickle
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/molecular-biology/promote
names = ['Class', 'id', 'Sequence']
data = pd.read_csv(url, names = names)
```

```
data.head(10)
```

	Class	id	Sequence
0	+	S10	\t\ttactagcaatacgcttgcgttcggtggttaagtatgtataat...
1	+	AMPC	\t\ttgctatcctgacagttgtcacgctgattggtgtcgttacaat...
2	+	AROH	\t\tgtactagagaactagtgcattagcttatttttttgttatcat...
3	+	DEOP2	\taattgtgatgtgtatcgaagtgtgttgcggagtagatgttagaa...
4	+	LEU1_TRNA	\ttcgataattaactattgacgaaaagctgaaaaccactagaatgc...
5	+	MALEFG	\taggggcaaggaggatggaaagaggttgccgtataaagaaactag...
6	+	MALK	\t\tcaggggtggaggatttaagccatctcctgatgacgcatagt...
7	+	RECA	\t\ttttctacaaaacacttgatactgtatgagcatacagtataat...
8	+	RPOB	\t\tcgacttaatatactgcgacaggacgtccgttctgtgtaaatc...
9	+	RRNAB_P1	\tttttaaatttcctcttgtcaggccggaataactccctataatgc...

```
data.shape
```

```
(106, 3)
```

```
data.dtypes
```

```
Class       object
id          object
Sequence    object
dtype: object
```

```
# Refining and structuring the data

# Build our dataset using custom pandas dataframe
classes = data.loc[:,'Class']
classes.head()
print()
print(classes.value_counts())
```

```
+    53
-    53
Name: Class, dtype: int64
```

```
# generate list of DNA sequence
sequence = list(data.loc[:, 'Sequence'])
sequence[-1]
```

```
'\t\ttaacattaataaataaggaggctctaatggcactcattagccaatcaatcaagaact'
```

```
#Remove tab from each sequence
dic = {}
for i, seq in enumerate(sequence):
    nucleotides = list(seq)
    nucleotides = [char for char in nucleotides if char != '\t']
    #append class assignment
    nucleotides.append(classes[i])

    dic[i] = nucleotides
list(dic[0])
```

```
['t',
 'a',
 'c',
 't',
 'a',
 'g',
 'c',
 'a',
 'a',
 't',
 'a',
 'c',
 'g',
 'c',
 't',
 't',
 'g',
 'c',
 'g',
 't',
 't',
 'c',
 'g',
 'g',
 't',
 'g',
```

```
 g ,
't',
't',
'a',
'a',
'g',
't',
'a',
't',
'g',
't',
'a',
't',
'a',
'a',
't',
'g',
'c',
'g',
'c',
'g',
'g',
'g',
'c',
't',
't',
'g',
't',
'c',
'g',
't',
'+']
```

```
# Convert Dict object into dataframe
df = pd.DataFrame(dic)
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	96	97	98	99	100	101	102	103	104	105
0	t	t	g	a	t	a	c	t	c	t	...	c	c	t	a	g	c	g	c	c	t
1	a	g	t	a	c	g	a	t	g	t	...	c	g	a	g	a	c	t	g	t	a
2	c	c	a	t	g	g	g	t	a	t	...	g	c	t	a	g	t	a	c	c	a
3	t	t	c	t	a	g	g	c	c	t	...	a	t	g	g	a	c	t	g	g	c
4	a	a	t	g	t	g	g	t	t	a	...	g	a	a	g	g	a	t	a	t	a

5 rows × 106 columns

```
# transpose dataframe into correct format
df = df.transpose()
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	57
0	t	a	c	t	a	g	c	a	a	t	...	g	c	t	t	g	t	c	g	t	+
1	t	g	c	t	a	t	c	c	t	g	...	c	a	t	c	g	c	c	a	a	+
2	g	t	a	c	t	a	g	a	g	a	...	c	a	c	c	c	g	g	c	g	+
3	a	a	t	t	g	t	g	a	t	g	...	a	a	c	a	a	a	c	t	c	+

```
# Rename the 57th column as it is our classes
df.rename(columns = {57:'Class'}, inplace = True)


df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
0	t	a	c	t	a	g	c	a	a	t	...	g	c	t	t	g	t	c	g	t	+
1	t	g	c	t	a	t	c	c	t	g	...	c	a	t	c	g	c	c	a	a	+
2	g	t	a	c	t	a	g	a	g	a	...	c	a	c	c	c	g	g	c	g	+
3	a	a	t	t	g	t	g	a	t	g	...	a	a	c	a	a	a	c	t	c	+
4	t	c	g	a	t	a	a	t	t	a	...	c	c	g	t	g	g	t	a	g	+

5 rows × 58 columns

```
temp = df.copy(deep=True)
temp = temp.drop(['Class'], axis = 1)

temp.head()
```

	0	1	2	3	4	5	6	7	8	9	...	47	48	49	50	51	52	53	54	55	56
0	t	a	c	t	a	g	c	a	a	t	...	g	g	c	t	t	g	t	c	g	t
1	t	g	c	t	a	t	c	c	t	g	...	g	c	a	t	c	g	c	c	a	a
2	g	t	a	c	t	a	g	a	g	a	...	c	c	a	c	c	c	g	g	c	g
3	a	a	t	t	g	t	g	a	t	g	...	t	a	a	c	a	a	a	c	t	c
4	t	c	g	a	t	a	a	t	t	a	...	t	c	c	g	t	g	g	t	a	g

5 rows × 57 columns

```
# Encoding using one-hot encoder:

enc = OneHotEncoder(handle_unknown='ignore')
enc.fit(temp)
print(enc.categories_)
df1 = enc.transform(temp).toarray()
del temp
# df1[1:3]
```

```
[array(['a', 'c', 'g', 't'], dtype=object), array(['a', 'c', 'g', 't'], dtype=object)
```

```
# Saving the one-hot encoder

with open("drive/MyDrive/EColi-encoder.pickle", "wb") as f:
    pickle.dump(enc, f)

# Loading the file later:
# encoder = pickle.load(f)
# data = encoder.transform(df).toarray()


df_new = pd.DataFrame(df1)
df_new.head()
```

	0	1	2	3	4	5	6	7	8	9	...	218	219	220	221	222	223
0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	1.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	1.0	0.0	0.0	0.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	...	1.0	0.0	0.0	1.0	0.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	1.0
4	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	0.0	1.0	1.0	0.0	0.0	0.0

5 rows × 228 columns

```
# Fixing the classes column:

df["Class"] = df["Class"].replace(to_replace =["+"], value =1)
df["Class"] = df["Class"].replace(to_replace =["-"], value =0)
df_new["Classes"] = df['Class']
df_new.head()
```

	0	1	2	3	4	5	6	7	8	9	...	219	220	221	222	223	224
0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	1.0	...	0.0	0.0	0.0	1.0	0.0	0.0
1	0.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0	...	0.0	1.0	0.0	0.0	0.0	1.0
2	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	...	0.0	0.0	1.0	0.0	0.0	0.0
3	1.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0	1.0	0.0	0.0	0.0	0.0	...	1.0	1.0	0.0	0.0	0.0	0.0

5 rows × 229 columns

```
#Encoding - Alternative
numerical_df = pd.get_dummies(df)
numerical_df.head()
```

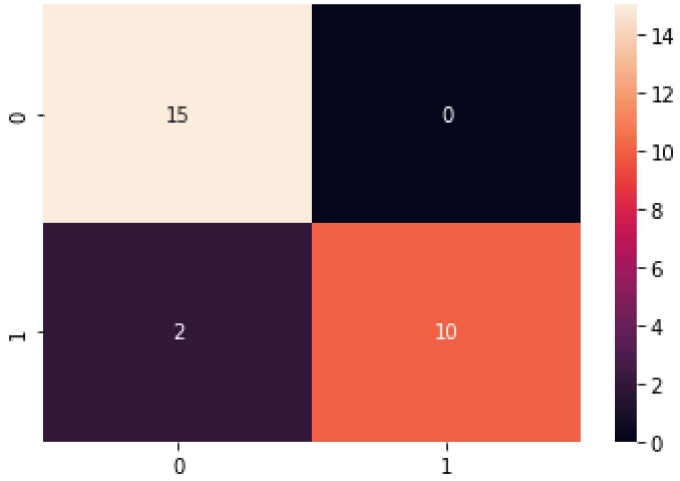
	Class	0_a	0_c	0_g	0_t	1_a	1_c	1_g	1_t	2_a	...	54_g	54_t	55_a	55_c
0	1	0	0	0	1	1	0	0	0	0	...	0	0	0	0
1	1	0	0	0	1	0	0	1	0	0	...	0	0	1	0
2	1	0	0	1	0	0	0	0	1	1	...	1	0	0	1
3	1	1	0	0	0	1	0	0	0	0	...	0	0	0	0
4	1	0	0	0	1	0	1	0	0	0	...	0	1	1	0

5 rows × 229 columns

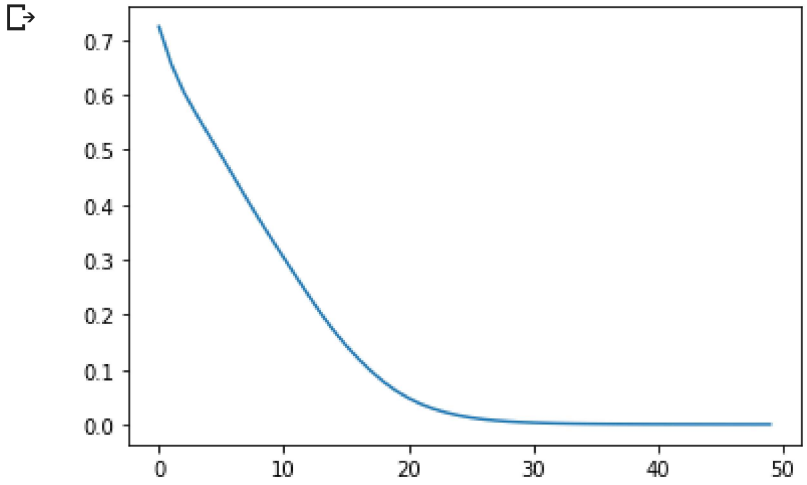
```
# Training and Testing the Classification Algorithms


y = df_new['Classes'].values# numerical_df['Class'].values
X = df_new.drop(['Classes'], axis = 1).values# numerical_df.drop(['Class'], axis = 1).valu

#define a seed for reproducibility
seed = 1

# Splitting data into training and testing data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state =


model = MLPClassifier(hidden_layer_sizes=(150,100,50), max_iter=300,activation = 'relu',so

model.fit(X_train, y_train)
print(model.score(X_train, y_train))

     1.0


#Predicting y for X_val
y_pred = model.predict(X_test)
model.score(X_test, y_test)

     0.9259259259259259


# Model evaluation
print(classification_report(y_test, y_pred))

                  precision    recall  f1-score   support

               0       1.00      0.88      0.94        17
               1       0.83      1.00      0.91        10

        accuracy                           0.93        27
       macro avg       0.92      0.94      0.92        27
    weighted avg       0.94      0.93      0.93        27



#Importing Confusion Matrix
#Comparing the predictions against the actual observations in y_val
```

```
cm = confusion_matrix(y_pred, y_test)
sns.heatmap(cm, annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f60b2bf0990>
```




```
# Plotting graph for MLPClassifier

loss_values = model.loss_curve_
plt.plot(loss_values)
plt.show()
```




```
# save the model to disk

filename = 'drive/MyDrive/E-Coli_model.pickle'
pickle.dump(model, open(filename, 'wb'))


genome = "ttactagcaatacgcttgcgttcggtggttaagtatgtataatgcgcgggcttgtcg"
# genome_false = "ttaacattaataaataaggaggctctaatggcactcattagccaatcaatcaagaac"
genome_list = list(genome)
print(genome_list)
df_test = pd.DataFrame(genome_list)
df_test = df_test.transpose()
```

```
['t', 't', 'a', 'c', 't', 'a', 'g', 'c', 'a', 'a', 't', 'a', 'c', 'g', 'c', 't', 't',
```

```
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	48	49	50	51	52	53	54	55	56	Class
0	t	a	c	t	a	g	c	a	a	t	...	g	c	t	t	g	t	c	g	t	1
1	t	g	c	t	a	t	c	c	t	g	...	c	a	t	c	g	c	c	a	a	1
2	g	t	a	c	t	a	g	a	g	a	...	c	a	c	c	c	g	g	c	g	1
3	a	a	t	t	g	t	g	a	t	g	...	a	a	c	a	a	a	c	t	c	1
4	t	c	g	a	t	a	a	t	t	a	...	c	c	g	t	g	g	t	a	g	1

5 rows × 58 columns

```
encoder = pickle.load(open("drive/MyDrive/EColi-encoder.pickle", 'rb'))
data_test = encoder.transform(df_test).toarray()
print(model.predict(data_test))
```

```
[1]
```

```
type(model.predict(data_test)[0])
```

```
numpy.int64
```

```
# load the model from disk
# loaded_model = pickle.load(open(filename, 'rb'))
# result = loaded_model.score(X_test, Y_test)
```