

ASSIGNMENT 2

Name : Pola Gnana Shekar

Roll No : 21CS10052

```
import pandas as pd

#import the data
train_data = pd.read_csv('./data_train.csv')
test_data = pd.read_csv('./data_test.csv')
output_data = pd.read_csv('./Copy of actual.csv')

#manipulating the data
train_data.drop(train_data.columns[0], axis=1, inplace=True)
test_data.drop(test_data.columns[0], axis=1, inplace=True)

num_columns_train = len(train_data.columns)
num_columns_test = len(test_data.columns)

for i in range(1, num_columns_train-1, 2):
    train_data[train_data.columns[i]] = train_data[train_data.columns[i+1]]

for i in range(1, num_columns_test-1, 2):
    test_data[test_data.columns[i]] = test_data[test_data.columns[i+1]]

train_data.drop(train_data.columns[2::2], axis=1, inplace=True)
test_data.drop(test_data.columns[2::2], axis=1, inplace=True)

train_data = train_data.T
test_data = test_data.T

train_data.reset_index(inplace=True)
train_data.columns = train_data.iloc[0]
train_data = train_data[1:]
train_data = train_data.rename(columns={'Gene Accession Number':
'patient'})

test_data.reset_index(inplace=True)
test_data.columns = test_data.iloc[0]
test_data = test_data[1:]
test_data = test_data.rename(columns={'Gene Accession Number':
'patient'})
```

```

train_data['patient'] = train_data['patient'].astype('int64')
test_data['patient'] = test_data['patient'].astype('int64')

trainD=pd.merge(train_data,output_data,on='patient',how='inner')
testD=pd.merge(test_data,output_data,on='patient',how='inner')

# The final data set has each patient as a sample data(rows) and each
# gene as on of it's feature(columns) and the last
# column represent the target value (cancer)
print("Train data: \n")
print(trainD)
print("Test data: \n")
print(testD)

```

Train data:

	patient	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at
0	1	A	A	A	
A					
1	2	A	A	A	
A					
2	3	A	A	A	
A					
3	4	A	A	A	
A					
4	5	A	A	A	
A					
5	6	A	A	A	
A					
6	7	A	A	A	
A					
7	8	A	A	A	
A					
8	9	A	A	A	
A					
9	10	A	A	A	
M					
10	11	A	A	A	
A					
11	12	A	A	A	
A					
12	13	A	A	A	
A					
13	14	A	A	A	
M					
14	15	A	A	A	
A					
15	16	A	A	A	

A					
16	17	A	A	A	
A					
17	18	A	A	A	
A					
18	19	A	A	A	
A					
19	20	A	A	A	
A					
20	21	A	A	A	
A					
21	22	A	A	A	
A					
22	23	A	A	A	
A					
23	24	A	A	A	
A					
24	25	A	A	A	
A					
25	26	A	A	A	
A					
26	27	A	A	A	
A					
27	34	A	A	A	
A					
28	35	A	A	A	
A					
29	36	A	A	A	
A					
30	37	A	A	A	
A					
31	38	A	A	A	
P					
32	28	A	A	A	
A					
33	29	A	A	A	
A					
34	30	A	A	A	
A					
35	31	A	A	A	
P					
36	32	A	A	A	
A					
37	33	A	A	A	
A					
	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at	\
0	A	A	A	A	
1	A	A	A	A	

2	A	A	A	A		
3	A	A	A	A		
4	A	A	A	A		
5	A	A	A	A		
6	A	A	A	A		
7	A	A	A	A		
8	A	A	A	A		
9	A	A	A	A		
10	A	A	A	A		
11	A	A	A	A		
12	A	A	A	A		
13	A	A	A	A		
14	A	A	A	A		
15	A	A	A	A		
16	A	A	A	A		
17	A	A	A	A		
18	A	A	A	A		
19	A	A	A	A		
20	A	A	A	A		
21	A	A	A	A		
22	A	A	A	A		
23	A	A	A	A		
24	A	A	A	A		
25	A	A	A	A		
26	A	A	A	A		
27	A	A	A	A		
28	A	A	A	A		
29	A	A	A	A		
30	A	A	A	A		
31	A	A	A	A		
32	A	A	A	A		
33	A	A	A	A		
34	A	A	A	A		
35	A	A	A	A		
36	A	A	A	A		
37	A	A	A	A		
AFFX-CreX-3_at ... U58516_at U73738_at X06956_at X16699_at						
X83863_at \	A	...	A	A	P	A
0	A	...	A	A	A	A
1	A	...	A	A	A	A
2	A	...	A	A	A	A
3	A	...	A	A	A	A
4	A	...	A	A	P	A
A						

5	A	...	A	A	A	A
A						
6	A	...	A	A	A	A
A						
7	A	...	A	A	A	A
A						
8	A	...	A	A	A	A
A						
9	A	...	P	A	A	A
A						
10	A	...	A	A	A	A
A						
11	A	...	A	A	A	A
A						
12	A	...	A	A	P	A
A						
13	A	...	A	A	A	A
A						
14	A	...	P	A	P	A
A						
15	A	...	A	A	A	A
A						
16	A	...	A	A	A	A
A						
17	A	...	A	A	P	A
A						
18	A	...	A	A	P	A
A						
19	A	...	A	A	A	A
A						
20	A	...	P	A	A	A
A						
21	A	...	A	A	P	A
A						
22	A	...	A	A	A	A
A						
23	A	...	A	A	A	A
A						
24	A	...	A	A	P	A
A						
25	A	...	A	A	P	A
A						
26	A	...	A	A	A	A
A						
27	A	...	A	A	A	A
A						
28	A	...	A	A	A	A
A						
29	A	...	A	A	P	A

A						
30	A	...	A	A	A	A
A						
31	A	...	A	A	A	A
A						
32	A	...	A	A	P	A
A						
33	A	...	A	A	P	A
A						
34	A	...	A	A	P	A
A						
35	A	...	A	A	A	A
A						
36	A	...	A	A	A	A
A						
37	A	...	A	A	A	A
A						

	Z17240_at	L49218_f_at	M71243_f_at	Z78285_f_at	cancer
0	A	A	A	A	ALL
1	A	A	A	A	ALL
2	P	A	A	A	ALL
3	A	A	A	A	ALL
4	A	A	A	A	ALL
5	A	A	A	A	ALL
6	A	A	P	A	ALL
7	P	A	P	A	ALL
8	A	P	A	A	ALL
9	P	A	M	A	ALL
10	A	A	A	A	ALL
11	A	A	A	A	ALL
12	A	A	A	A	ALL
13	A	A	A	A	ALL
14	A	A	A	A	ALL
15	P	A	P	A	ALL
16	P	A	A	A	ALL
17	P	A	A	A	ALL
18	A	A	A	A	ALL
19	A	A	P	A	ALL
20	A	A	A	A	ALL
21	A	A	M	A	ALL
22	A	A	M	A	ALL
23	A	A	A	A	ALL
24	A	A	P	A	ALL
25	P	A	A	A	ALL
26	A	A	A	A	ALL
27	A	A	M	A	AML
28	A	A	P	A	AML
29	A	A	A	A	AML

30	A	A	P	A	AML
31	A	A	P	A	AML
32	A	A	P	A	AML
33	A	A	P	A	AML
34	A	A	A	A	AML
35	A	A	A	A	AML
36	A	A	A	A	AML
37	A	A	P	A	AML

[38 rows x 7131 columns]

Test data:

	patient	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at \
0	39	A	A	A	
A					
1	40	A	A	A	
A					
2	42	A	A	A	
A					
3	47	A	A	A	
A					
4	48	A	A	A	
A					
5	49	A	A	A	
A					
6	41	A	A	A	
A					
7	43	A	A	A	
A					
8	44	A	A	A	
A					
9	45	A	A	A	
A					
10	46	A	A	A	
A					
11	70	A	A	A	
A					
12	71	A	A	A	
A					
13	72	A	A	A	
A					
14	68	A	A	A	
A					
15	69	A	A	A	
A					
16	67	A	A	A	
A					
17	55	A	A	A	

A				
18	56	A	A	A
A				
19	59	A	A	A
A				
20	52	A	A	A
A				
21	53	A	A	A
A				
22	51	A	A	A
A				
23	50	A	A	A
A				
24	54	A	A	A
P				
25	57	A	A	A
A				
26	58	A	A	A
A				
27	60	A	A	A
A				
28	61	A	A	A
A				
29	65	A	A	A
A				
30	66	A	A	A
A				
31	63	A	A	A
A				
32	64	A	A	A
A				
33	62	A	A	A
A				
	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at \
0	A	A	A	A
1	A	A	A	A
2	A	A	A	A
3	A	A	A	A
4	A	A	A	A
5	A	A	A	A
6	A	A	A	A
7	A	A	A	A
8	A	A	A	A
9	A	A	A	A
10	A	A	A	A
11	A	A	A	A
12	A	A	A	A
13	A	A	A	A

14	A	A	A	A
15	A	A	A	A
16	A	A	A	A
17	A	A	A	A
18	A	A	A	A
19	A	A	A	A
20	A	A	A	A
21	A	A	A	A
22	A	A	A	A
23	A	A	A	A
24	A	A	A	A
25	A	A	A	A
26	A	A	A	A
27	A	A	A	A
28	A	A	A	A
29	A	A	A	A
30	A	A	A	A
31	A	A	A	A
32	A	A	A	A
33	A	A	A	A

AFFX-CreX-3_at ... U58516_at U73738_at X06956_at X16699_at
X83863_at \

0	A	...	A	A	A	A
A						
1	A	...	A	A	P	A
A						
2	A	...	A	A	P	A
P						
3	A	...	A	A	A	A
A						
4	A	...	A	A	P	A
A						
5	A	...	A	A	A	A
A						
6	A	...	P	A	A	A
A						
7	A	...	A	A	P	A
A						
8	A	...	P	A	P	A
A						
9	A	...	A	A	A	A
A						
10	A	...	P	A	A	A
A						
11	A	...	A	A	A	A
A						
12	A	...	A	A	P	A
A						

13	A	...	A	A	P	A
A						
14	A	...	A	A	P	A
A						
15	A	...	A	A	P	A
A						
16	A	...	A	A	A	A
A						
17	A	...	A	A	A	A
A						
18	A	...	A	A	P	A
A						
19	A	...	A	A	P	A
A						
20	A	...	A	A	A	A
A						
21	A	...	A	A	A	A
A						
22	A	...	A	A	A	A
A						
23	A	...	A	A	A	A
A						
24	A	...	A	A	A	A
A						
25	A	...	A	A	A	A
A						
26	A	...	A	A	A	A
A						
27	A	...	A	A	A	A
A						
28	A	...	A	A	A	A
A						
29	A	...	A	A	P	A
A						
30	A	...	A	A	P	A
A						
31	A	...	A	A	P	A
A						
32	A	...	A	A	A	A
A						
33	A	...	A	A	P	A
A						

	Z17240_at	L49218_f_at	M71243_f_at	Z78285_f_at	cancer
0	A	A	A	A	ALL
1	A	A	A	A	ALL
2	A	A	P	A	ALL
3	A	A	P	A	ALL
4	A	A	A	A	ALL

5	A	A	A	A	ALL
6	A	A	A	A	ALL
7	A	A	P	A	ALL
8	A	A	A	A	ALL
9	A	A	A	A	ALL
10	A	A	A	A	ALL
11	A	A	P	P	ALL
12	A	A	A	A	ALL
13	P	A	A	A	ALL
14	A	A	A	A	ALL
15	A	A	A	A	ALL
16	A	A	A	A	ALL
17	A	A	A	A	ALL
18	A	A	P	A	ALL
19	A	A	A	A	ALL
20	A	A	A	A	AML
21	P	A	A	A	AML
22	A	A	P	A	AML
23	A	A	P	A	AML
24	A	A	A	A	AML
25	A	A	A	A	AML
26	P	A	A	A	AML
27	A	A	M	A	AML
28	A	A	A	A	AML
29	A	A	A	A	AML
30	A	A	P	A	AML
31	A	A	A	A	AML
32	A	A	A	A	AML
33	A	A	A	A	AML

[34 rows x 7131 columns]

#encoding the data values to make it processable to train the models

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

Split the data into features (X) and target labels (y)

```
X_train = trainD.drop(['patient', 'cancer'], axis=1)
y_train = trainD['cancer']
```

```
X_test = testD.drop(['patient', 'cancer'], axis=1)
y_test = testD['cancer']
```

Initialize the LabelEncoder

```
encoder = LabelEncoder()
```

Concatenate X_train and X_test to ensure all categories are seen

```
combined_data = pd.concat([X_train, X_test], axis=0)
```

Apply label encoding to combined_data

```

for column in combined_data.columns:
    combined_data[column] =
encoder.fit_transform(combined_data[column])

# Split the encoded data back into X_train_encoded and X_test_encoded
X_train_encoded = combined_data.iloc[:len(X_train)]
X_test_encoded = combined_data.iloc[len(X_train):]

# Print X_train_encoded
print("X_train_encoded:\n")
print(X_train_encoded)

# Print X_test_encoded
print("\nX_test_encoded:\n")
print(X_test_encoded)

```

X_train_encoded:

	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	
8	0	0	0	0	
9	0	0	0	1	
10	0	0	0	0	
11	0	0	0	0	
12	0	0	0	0	
13	0	0	0	1	
14	0	0	0	0	
15	0	0	0	0	
16	0	0	0	0	
17	0	0	0	0	
18	0	0	0	0	
19	0	0	0	0	
20	0	0	0	0	
21	0	0	0	0	
22	0	0	0	0	
23	0	0	0	0	
24	0	0	0	0	
25	0	0	0	0	
26	0	0	0	0	
27	0	0	0	0	
28	0	0	0	0	
29	0	0	0	0	
30	0	0	0	0	

31	0	0	0	2
32	0	0	0	0
33	0	0	0	0
34	0	0	0	0
35	0	0	0	2
36	0	0	0	0
37	0	0	0	0
AFFX-BioC-3_at AFFX-BioDn-5_at AFFX-BioDn-3_at AFFX-CreX-5_at \				
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0

21	0	0	0	0
22	0	0	0	0
23	0	0	0	0
24	0	0	0	0
25	0	0	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
34	0	0	0	0
35	0	0	0	0
36	0	0	0	0
37	0	0	0	0

	AFFX-CreX-3_at	AFFX-BioB-5_st	...	U48730_at	U58516_at
U73738_at \					
0	0	0	...	0	0
0					
1	0	0	...	0	0
0					
2	0	0	...	1	0
0					
3	0	0	...	0	0
0					
4	0	0	...	1	0
0					
5	0	0	...	0	0
0					
6	0	0	...	0	0

0					
7	0	0	...	0	0
0					
8	0	0	...	1	0
0					
9	0	0	...	0	1
0					
10	0	0	...	1	0
0					
11	0	0	...	0	0
0					
12	0	0	...	1	0
0					
13	0	0	...	1	0
0					
14	0	0	...	0	1
0					
15	0	0	...	0	0
0					
16	0	0	...	1	0
0					
17	0	0	...	1	0
0					
18	0	0	...	0	0
0					
19	0	0	...	1	0
0					
20	0	0	...	0	1
0					
21	0	0	...	1	0
0					
22	0	0	...	0	0
0					
23	0	0	...	0	0
0					
24	0	0	...	0	0
0					
25	0	0	...	1	0
0					
26	0	0	...	0	0
0					
27	0	0	...	0	0
0					
28	0	0	...	0	0
0					
29	0	0	...	1	0
0					
30	0	0	...	0	0
0					

31	0	0	...	0	0
0					
32	0	0	...	1	0
0					
33	0	0	...	0	0
0					
34	0	0	...	1	0
0					
35	0	0	...	0	0
0					
36	0	0	...	1	0
0					
37	0	0	...	0	0
0					

	X06956_at	X16699_at	X83863_at	Z17240_at	L49218_f_at
M71243_f_at \					
0	1	0	0	0	0
0					
1	0	0	0	0	0
0					
2	0	0	0	1	0
0					
3	0	0	0	0	0
0					
4	1	0	0	0	0
0					
5	0	0	0	0	0
0					
6	0	0	0	0	0
2					
7	0	0	0	1	0
2					
8	0	0	0	0	1
0					
9	0	0	0	1	0
1					
10	0	0	0	0	0
0					
11	0	0	0	0	0
0					
12	1	0	0	0	0
0					
13	0	0	0	0	0
0					
14	1	0	0	0	0
0					
15	0	0	0	1	0
2					

16	0	0	0	1	0
0					
17	1	0	0	1	0
0					
18	1	0	0	0	0
0					
19	0	0	0	0	0
2					
20	0	0	0	0	0
0					
21	1	0	0	0	0
1					
22	0	0	0	0	0
1					
23	0	0	0	0	0
0					
24	1	0	0	0	0
2					
25	1	0	0	1	0
0					
26	0	0	0	0	0
0					
27	0	0	0	0	0
1					
28	0	0	0	0	0
2					
29	1	0	0	0	0
0					
30	0	0	0	0	0
2					
31	0	0	0	0	0
2					
32	1	0	0	0	0
2					
33	1	0	0	0	0
2					
34	1	0	0	0	0
0					
35	0	0	0	0	0
0					
36	0	0	0	0	0
0					
37	0	0	0	0	0
2					

Z78285_f_at

0	0
1	0
2	0

3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0
34	0
35	0
36	0
37	0

[38 rows x 7129 columns]

X_test_encoded:

	AFFX-BioB-5_at	AFFX-BioB-M_at	AFFX-BioB-3_at	AFFX-BioC-5_at	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	
5	0	0	0	0	
6	0	0	0	0	
7	0	0	0	0	
8	0	0	0	0	

9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	0	0
22	0	0	0	0
23	0	0	0	0
24	0	0	0	2
25	0	0	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
	AFFX-BioC-3_at	AFFX-BioDn-5_at	AFFX-BioDn-3_at	AFFX-CreX-5_at \
0	0	0	0	0
1	0	0	0	0
2	0	0	0	0
3	0	0	0	0
4	0	0	0	0
5	0	0	0	0
6	0	0	0	0
7	0	0	0	0
8	0	0	0	0
9	0	0	0	0
10	0	0	0	0

11	0	0	0	0
12	0	0	0	0
13	0	0	0	0
14	0	0	0	0
15	0	0	0	0
16	0	0	0	0
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	0	0	0	0
21	0	0	0	0
22	0	0	0	0
23	0	0	0	0
24	0	0	0	0
25	0	0	0	0
26	0	0	0	0
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
30	0	0	0	0
31	0	0	0	0
32	0	0	0	0
33	0	0	0	0
AFFX-CreX-3_at AFFX-BioB-5_st ... U48730_at U58516_at				
U73738_at \	0	0	...	0
0				0
0				

1	0	0	...	0	0
0					
2	0	0	...	1	0
0					
3	0	0	...	0	0
0					
4	0	0	...	1	0
0					
5	0	0	...	0	0
0					
6	0	0	...	0	1
0					
7	0	0	...	0	0
0					
8	0	0	...	0	1
0					
9	0	0	...	0	0
0					
10	0	0	...	0	1
0					
11	0	0	...	1	0
0					
12	0	0	...	1	0
0					
13	0	0	...	0	0
0					
14	0	0	...	1	0
0					
15	0	0	...	1	0
0					
16	0	0	...	1	0
0					
17	0	0	...	0	0
0					
18	0	0	...	0	0
0					
19	0	0	...	0	0
0					
20	0	0	...	0	0
0					
21	0	0	...	0	0
0					
22	0	0	...	0	0
0					
23	0	0	...	0	0
0					
24	0	0	...	0	0
0					
25	0	0	...	0	0

0					
26	0	0	...	0	0
0					
27	0	0	...	0	0
0					
28	0	0	...	1	0
0					
29	0	0	...	0	0
0					
30	0	0	...	0	0
0					
31	0	0	...	0	0
0					
32	0	0	...	0	0
0					
33	0	0	...	0	0
0					

	X06956_at	X16699_at	X83863_at	Z17240_at	L49218_f_at
M71243_f_at \					
0	0	0	0	0	0
0					
1	1	0	0	0	0
0					
2	1	0	1	0	0
2					
3	0	0	0	0	0
2					
4	1	0	0	0	0
0					
5	0	0	0	0	0
0					
6	0	0	0	0	0
0					
7	1	0	0	0	0
2					
8	1	0	0	0	0
0					
9	0	0	0	0	0
0					
10	0	0	0	0	0
0					
11	0	0	0	0	0
2					
12	1	0	0	0	0
0					
13	1	0	0	1	0
0					
14	1	0	0	0	0
0					

15	1	0	0	0	0
0					
16	0	0	0	0	0
0					
17	0	0	0	0	0
0					
18	1	0	0	0	0
2					
19	1	0	0	0	0
0					
20	0	0	0	0	0
0					
21	0	0	0	1	0
0					
22	0	0	0	0	0
2					
23	0	0	0	0	0
2					
24	0	0	0	0	0
0					
25	0	0	0	0	0
0					
26	0	0	0	1	0
0					
27	0	0	0	0	0
1					
28	0	0	0	0	0
0					
29	1	0	0	0	0
0					
30	1	0	0	0	0
2					
31	1	0	0	0	0
0					
32	0	0	0	0	0
0					
33	1	0	0	0	0
0					

	Z78285_f_at
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0

9	0
10	0
11	1
12	0
13	0
14	0
15	0
16	0
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	0
27	0
28	0
29	0
30	0
31	0
32	0
33	0

[34 rows x 7129 columns]

#SVM model

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix
```

Define a list of kernel names

```
kernels = ['linear', 'rbf', 'poly', 'sigmoid']
```

```
for kernel in kernels:
```

Create an SVM model with the specified kernel

```
model = SVC(kernel=kernel)
```

Scale the features

```
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train_encoded)
```

```
X_test_scaled = scaler.transform(X_test_encoded)
```

Train the model

```
model.fit(X_train_scaled, y_train)
```

Make predictions on the test data


```

y_pred = model.predict(X_test_scaled)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, pos_label='ALL')
recall = recall_score(y_test, y_pred, pos_label='ALL')
f1 = f1_score(y_test, y_pred, pos_label='ALL')
confusion = confusion_matrix(y_test, y_pred)

# Print the evaluation metrics
print(f"SVM Model with {kernel} kernel")
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
print(f"Confusion Matrix:\n{confusion}\n")

```

```

SVM Model with linear kernel
Accuracy: 0.9117647058823529
Precision: 0.8695652173913043
Recall: 1.0
F1 Score: 0.9302325581395349
Confusion Matrix:
[[20  0]
 [ 3 11]]

```

```

SVM Model with rbf kernel
Accuracy: 0.5882352941176471
Precision: 0.5882352941176471
Recall: 1.0
F1 Score: 0.7407407407407407
Confusion Matrix:
[[20  0]
 [14  0]]

```

```

SVM Model with poly kernel
Accuracy: 0.5882352941176471
Precision: 0.5882352941176471
Recall: 1.0
F1 Score: 0.7407407407407407
Confusion Matrix:
[[20  0]
 [14  0]]

```

```

SVM Model with sigmoid kernel
Accuracy: 0.8529411764705882
Precision: 0.8
Recall: 1.0
F1 Score: 0.8888888888888889
Confusion Matrix:

```

```
[[20  0]
 [ 5  9]]
```

```
#Random Forest
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix
```

```
# Create a Random Forest model
```

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
# Train the model
```

```
rf_model.fit(X_train_encoded, y_train)
```

```
# Make predictions on the test data
```

```
y_pred_rf = rf_model.predict(X_test_encoded)
```

```
# Evaluate the model
```

```
accuracy_rf = accuracy_score(y_test, y_pred_rf)
precision_rf = precision_score(y_test, y_pred_rf, pos_label='ALL')
recall_rf = recall_score(y_test, y_pred_rf, pos_label='ALL')
f1_rf = f1_score(y_test, y_pred_rf, pos_label='ALL')
confusion_rf = confusion_matrix(y_test, y_pred_rf)
```

```
# Print the evaluation metrics
```

```
print("Random Forest Model")
print(f"Accuracy: {accuracy_rf}")
print(f"Precision: {precision_rf}")
print(f"Recall: {recall_rf}")
print(f"F1 Score: {f1_rf}")
print(f"Confusion Matrix:\n{confusion_rf}\n")
```

```
Random Forest Model
```

```
Accuracy: 0.8823529411764706
```

```
Precision: 0.8333333333333334
```

```
Recall: 1.0
```

```
F1 Score: 0.9090909090909091
```

```
Confusion Matrix:
```

```
[[20  0]
 [ 4 10]]
```

```
from itertools import product
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, confusion_matrix
```

```
# Define the hyperparameter grid to search through
```

```
param_grid = {
```

```

    'hidden_layer_sizes': [(100, 50), (50, 75), (200, 100)],
    'alpha': [0.001, 0.01],
    'max_iter': [500, 1000],
    'solver': ['adam', 'lbfgs'],
    'learning_rate_init': [0.001, 0.01]
}

# Create a list of all parameter combinations
param_combinations = list(product(param_grid['hidden_layer_sizes'],
                                  param_grid['alpha'], param_grid['max_iter'],
                                  param_grid['solver'],
                                  param_grid['learning_rate_init']))

# Create a Neural Network model
nn_model = MLPClassifier()

# Scale the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train_encoded)
X_test_scaled = scaler.transform(X_test_encoded)

best_accuracy = 0
best_params = None
best_precision = 0
best_recall = 0
best_f1_score = 0
best_confusion_matrix = None

# Iterate through each parameter combination
for params in param_combinations:
    hidden_layer_sizes, alpha, max_iter, solver, learning_rate_init =
    params

    # Print the current parameter combination being processed
    print(f"Working on {params}...")

    # Set the hyperparameters for the current combination
    nn_model.set_params(hidden_layer_sizes=hidden_layer_sizes,
                        alpha=alpha, max_iter=max_iter, solver=solver,
                        learning_rate_init=learning_rate_init)

    # Train the model
    nn_model.fit(X_train_scaled, y_train)

    # Make predictions on the test data
    y_pred_nn = nn_model.predict(X_test_scaled)

    # Evaluate the model
    accuracy_nn = accuracy_score(y_test, y_pred_nn)

```

```

# Check if this combination has the best accuracy so far
if accuracy_nn > best_accuracy:
    best_accuracy = accuracy_nn
    best_params = params

# Calculate precision, recall, and F1-score
precision_nn = precision_score(y_test, y_pred_nn,
pos_label='ALL')
recall_nn = recall_score(y_test, y_pred_nn, pos_label='ALL')
f1_nn = f1_score(y_test, y_pred_nn, pos_label='ALL')

# Calculate confusion matrix
confusion_nn = confusion_matrix(y_test, y_pred_nn)

# Update best metrics
best_precision = precision_nn
best_recall = recall_nn
best_f1_score = f1_nn
best_confusion_matrix = confusion_nn

# Print the best result and its hyperparameters
print("\n\nBest Neural Network Model")
print(f"Best Hyperparameters: {best_params}")
print(f"Best Accuracy: {best_accuracy}")
print(f"Best Precision: {best_precision}")
print(f"Best Recall: {best_recall}")
print(f"Best F1 Score: {best_f1_score}")
print("Best Confusion Matrix:")
print(best_confusion_matrix)

```

```

Working on ((100, 50), 0.001, 500, 'adam', 0.001)...
Working on ((100, 50), 0.001, 500, 'adam', 0.01)...
Working on ((100, 50), 0.001, 500, 'lbfgs', 0.001)...
Working on ((100, 50), 0.001, 500, 'lbfgs', 0.01)...
Working on ((100, 50), 0.001, 1000, 'adam', 0.001)...
Working on ((100, 50), 0.001, 1000, 'adam', 0.01)...
Working on ((100, 50), 0.001, 1000, 'lbfgs', 0.001)...
Working on ((100, 50), 0.001, 1000, 'lbfgs', 0.01)...
Working on ((100, 50), 0.01, 500, 'adam', 0.001)...
Working on ((100, 50), 0.01, 500, 'adam', 0.01)...
Working on ((100, 50), 0.01, 500, 'lbfgs', 0.001)...
Working on ((100, 50), 0.01, 500, 'lbfgs', 0.01)...
Working on ((100, 50), 0.01, 1000, 'adam', 0.001)...
Working on ((100, 50), 0.01, 1000, 'adam', 0.01)...
Working on ((100, 50), 0.01, 1000, 'lbfgs', 0.001)...
Working on ((100, 50), 0.01, 1000, 'lbfgs', 0.01)...
Working on ((50, 75), 0.001, 500, 'adam', 0.001)...
Working on ((50, 75), 0.001, 500, 'adam', 0.01)...
Working on ((50, 75), 0.001, 500, 'lbfgs', 0.001)...
Working on ((50, 75), 0.001, 500, 'lbfgs', 0.01)...

```

```
Working on ((50, 75), 0.001, 1000, 'adam', 0.001)...
Working on ((50, 75), 0.001, 1000, 'adam', 0.01)...
Working on ((50, 75), 0.001, 1000, 'lbfgs', 0.001)...
Working on ((50, 75), 0.001, 1000, 'lbfgs', 0.01)...
Working on ((50, 75), 0.01, 500, 'adam', 0.001)...
Working on ((50, 75), 0.01, 500, 'adam', 0.01)...
Working on ((50, 75), 0.01, 500, 'lbfgs', 0.001)...
Working on ((50, 75), 0.01, 500, 'lbfgs', 0.01)...
Working on ((50, 75), 0.01, 1000, 'adam', 0.001)...
Working on ((50, 75), 0.01, 1000, 'adam', 0.01)...
Working on ((50, 75), 0.01, 1000, 'lbfgs', 0.001)...
Working on ((50, 75), 0.01, 1000, 'lbfgs', 0.01)...
Working on ((200, 100), 0.001, 500, 'adam', 0.001)...
Working on ((200, 100), 0.001, 500, 'adam', 0.01)...
Working on ((200, 100), 0.001, 500, 'lbfgs', 0.001)...
Working on ((200, 100), 0.001, 500, 'lbfgs', 0.01)...
Working on ((200, 100), 0.001, 1000, 'adam', 0.001)...
Working on ((200, 100), 0.001, 1000, 'adam', 0.01)...
Working on ((200, 100), 0.001, 1000, 'lbfgs', 0.001)...
Working on ((200, 100), 0.001, 1000, 'lbfgs', 0.01)...
Working on ((200, 100), 0.01, 500, 'adam', 0.001)...
Working on ((200, 100), 0.01, 500, 'adam', 0.01)...
Working on ((200, 100), 0.01, 500, 'lbfgs', 0.001)...
Working on ((200, 100), 0.01, 500, 'lbfgs', 0.01)...
Working on ((200, 100), 0.01, 1000, 'adam', 0.001)...
Working on ((200, 100), 0.01, 1000, 'adam', 0.01)...
Working on ((200, 100), 0.01, 1000, 'lbfgs', 0.001)...
Working on ((200, 100), 0.01, 1000, 'lbfgs', 0.01)...
```

Best Neural Network Model

Best Hyperparameters: ((200, 100), 0.001, 1000, 'adam', 0.001)

Best Accuracy: 1.0

Best Precision: 1.0

Best Recall: 1.0

Best F1 Score: 1.0

Best Confusion Matrix:

```
[[20  0]
 [ 0 14]]
```