

```

from flask import Flask, request, jsonify
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
from sklearn import metrics
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import time

```

```
app = Flask(__name__)
```

```
df = pd.read_csv("dataset5.csv")
```

```
df
```

	payload	label
0	benalmdena	norm
1	1%")) or updatexml(1808,concat(0x2e,0x7171706a...	anom
2	-4882' order by 1#	anom
3	SELECT COUNT (boneID) FROM blank	norm
4	2.61522E+15	norm
...
66261	Nelsonstad	norm
66262	Employee hear place imagine purpose imagine jo...	norm
66263	1988-05-07	norm
66264	4917992725039509	norm
66265	West Cindymouth	norm

```
[66266 rows x 2 columns]
```

```

print("\n-----Sample-----\n")
print(df.sample(10))
print("\n-----Dataset Information-----\n")
print(df.info())
print("\n-----Dataset Statistics-----\n")
print(df.describe())
print("\n-----Dataset Size-----\n")
print(df.shape)
print("\n-----Number of Null Values in the
Dataset-----\n")
print(df.isnull().sum())

```

```
-----Sample-----
```

		payload	label
1912	SELECT AVG (setting) FROM than		norm
21984		mikel	norm
21383		arenys de mar	norm
15920	SELECT * FROM passage WHERE by NOT LIKE '[lai...		norm
4285	1%" union all select null,null,null,null,null,...		anom
36032	1') as bfgl where 8959 = 8959 union all ...		anom
45036		calle valsendero, 48,	norm
46707	SELECT modern AS forty, belong AS [done] FR...		norm
20108	1')) procedure analyse(extractvalue(9627,conca...		anom
22355	1' in boolean mode) union all select null,null...		anom

-----Dataset Information-----

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 66266 entries, 0 to 66265
Data columns (total 2 columns):
#   Column      Non-Null Count  Dtype
---  ---
0    payload    66266 non-null   object
1    label      66266 non-null   object
dtypes: object(2)
memory usage: 1.0+ MB
None
```

-----Dataset Statistics-----

	payload	label
count	66266	66266
unique	58361	2
top	#NAME?	norm
freq	20	43160

-----Dataset Size-----

```
(66266, 2)
```

-----Number of Null Values in the Dataset-----

```
payload    0
label      0
dtype: int64
```

```
# Histogram of text lengths
```

```
df['text_length'] = df['payload'].apply(len)
```

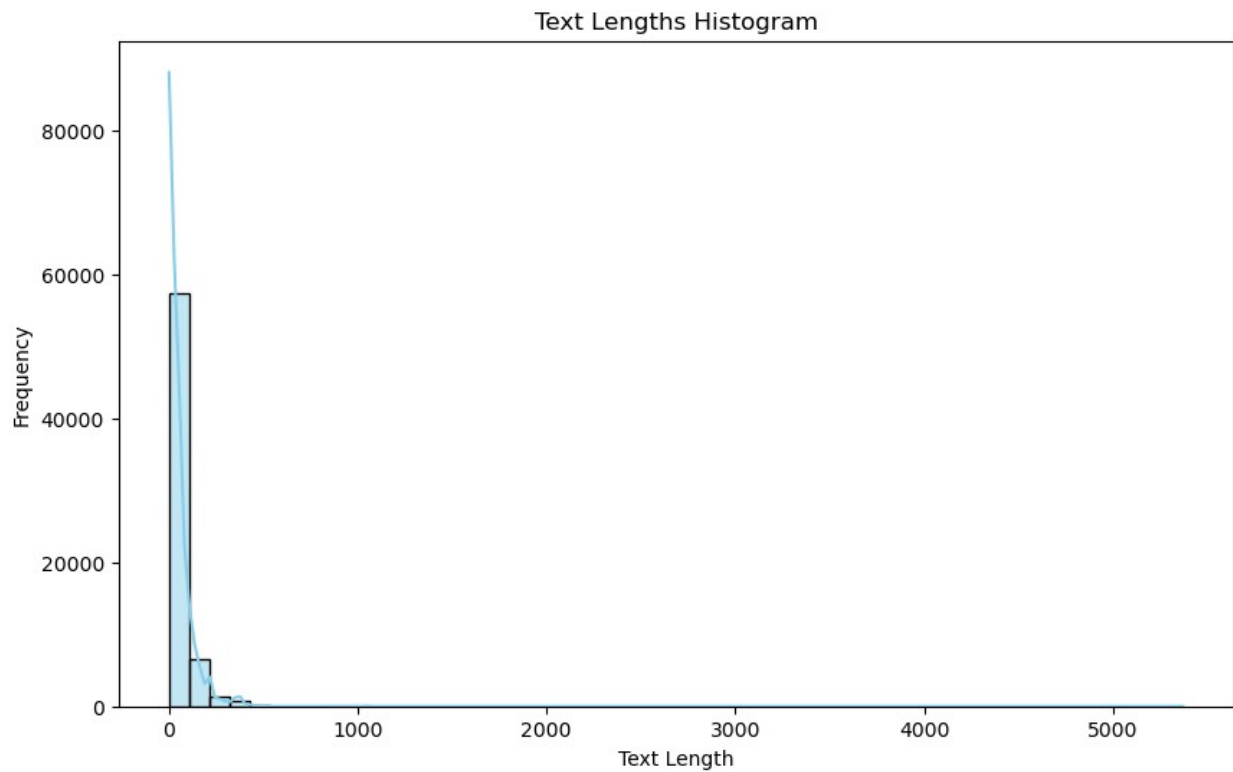
```
plt.figure(figsize=(10, 6))
```

```
sns.histplot(df['text_length'], bins=50, kde=True, color='skyblue')
```

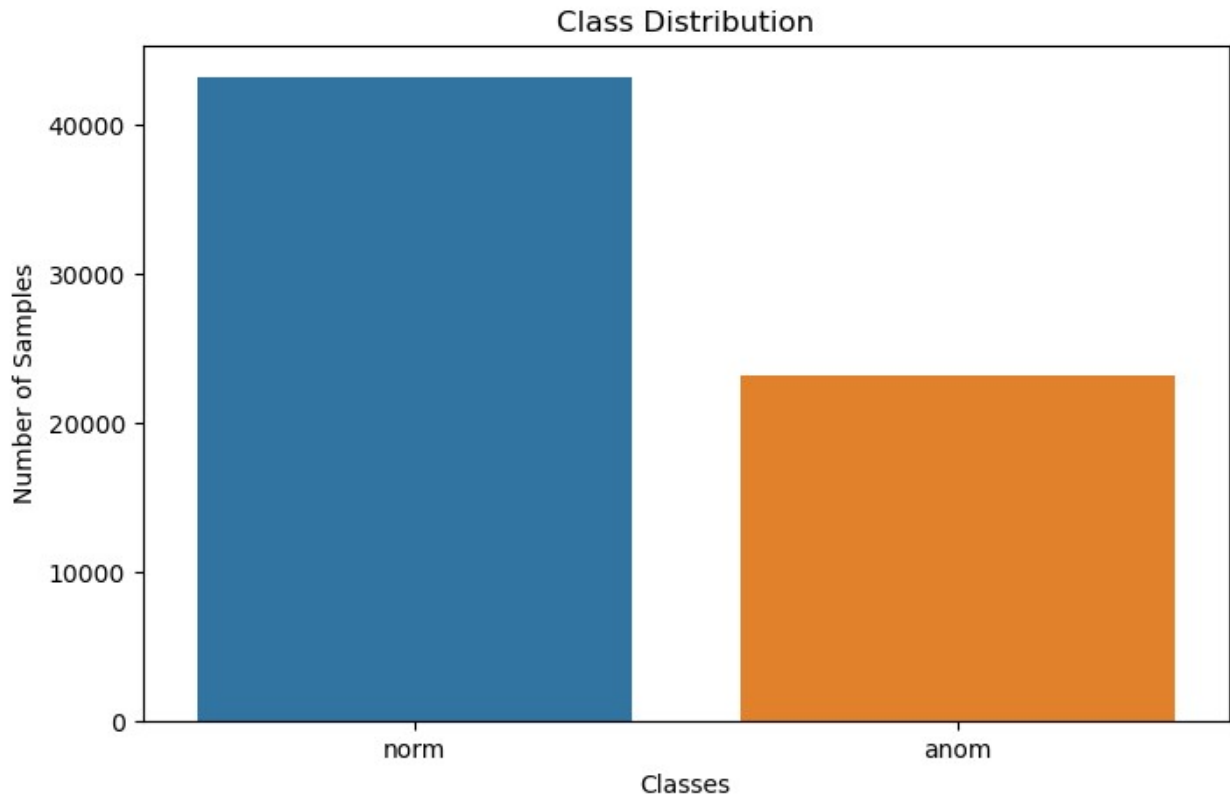
```
plt.title('Text Lengths Histogram')
```

```
plt.xlabel('Text Length')
```

```
plt.ylabel('Frequency')  
plt.show()
```



```
# A bar chart showing class distribution  
plt.figure(figsize=(8, 5))  
sns.countplot(x='label', data=df)  
plt.title('Class Distribution')  
plt.xlabel('Classes')  
plt.ylabel('Number of Samples')  
plt.show()
```



```
print(df.head(50))
```

	payload	label
text_length		
0	benalmdena	norm
10		
1	1%")) or updatexml(1808,concat(0x2e,0x7171706a...	anom
108		
2	-4882' order by 1#	anom
18		
3	SELECT COUNT (boneID) FROM blank	norm
34		
4	2.61522E+15	norm
11		
5	50491	norm
5		
6	boom@paseotuperro.mo	norm
20		
7	1' in boolean mode);select count(*) from sysib...	anom
110		
8	select like ('abcdefg',upper (hex (ra...	anom
116		
9	1"));begin dbms_lock.sleep(5); end and (("rqay...	anom
58		
10	SELECT TOP 50 PERCENT * FROM stop SELECT * FR0...	norm

```

117
11      DELETE FROM rubber WHERE how = 'dirty'  norm
38
12      11690  norm
5
13      26562587w  norm
9
14      juan  norm
4
15      kental  norm
6
16      1') or (select * from (select(sleep(5)))sddo)#  anom
46
17      SELECT * FROM shake WHERE ahead IN  ( SELECT e...  norm
66
18      5130  norm
4
19      breceda perdig  norm
14
20      1')) and 3202=like('abcdefg',upper(hex(randomb...  anom
84
21      9.63962E+15  norm
11
22      quilolitr-o  norm
11
23      1');begin user_lock.sleep(5); end  anom
33
24      promanar  norm
8
25      96399738e  norm
9
26      09352334m  norm
9
27      /aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa...  anom
291
28      carstens  norm
8
29      1" or 8315 =  ( select count ( * ) fr...  anom
122
30      1';begin user_lock.sleep(5); end and 'blit'='blit  anom
49
31      SELECT * FROM  ( SELECT worth FROM foreign )  norm
45
32      1) union all select null#  anom
25
33      7681  norm
4
34      1.83452E+15  norm
11

```

```

35          oli4c5r  norm
7
36          manjarrrs  norm
8
37          â( or 1 = 1 --  anom
17
38  1')))) or elt(6272=6272,sleep(5)) and (((('ayjb'...  anom
57
39          villanueva de cauche 169,  norm
25
40          5.01E+15  norm
8
41  select * from users where id = 1 +1 union s...  anom
102
42          drayton  norm
7
43  -6947')) as qqjp where 6700=6700 union all sel...  anom
55
44          9.05465E+15  norm
11
45  SELECT COUNT ( voyage ) , idea FROM sad BY ...  norm
50
46  SELECT club ( s ) FROM drawn SELECT roar ( s )  norm
48
47  SELECT * FROM birthday WHERE all = 'party' LIM...  norm
50
48          frydtberg@archivodeimagenes.cat  norm
31
49          or '*'  anom
8

```

Split the data into training and testing sets

```

X_train, X_test, y_train, y_test = train_test_split(df['payload'],
df['label'], test_size=0.2, random_state=42)

```

Create the model

```

start_time = time.time()
model1 = make_pipeline(TfidfVectorizer(), MultinomialNB())
model1.fit(X_train, y_train)
end_time = time.time()
elapsed_time1 = end_time - start_time
print("\n-----MultinomialNB Models-----\n")
print("education time: ", elapsed_time1)
print("Training data score: ", model1.score(X_train, y_train))

```

-----MultinomialNB Models-----

```

education time:  0.7755253314971924
Training data score:  0.9912472647702407

```

```

start_time = time.time()
model2 = make_pipeline(TfidfVectorizer(),
LogisticRegression(max_iter=1000))
model2.fit(X_train, y_train)
end_time = time.time()
elapsed_time2 = end_time - start_time
print("\n-----LogisticRegression Model-----\n")
print("education time: ", elapsed_time2)
print("Training data score: ", model2.score(X_train, y_train))

-----LogisticRegression Model-----

education time:  1.234677791595459
Training data score:  0.9813061193691994

start_time = time.time()
model3 = make_pipeline(TfidfVectorizer(), SVC(kernel='linear'))
model3.fit(X_train, y_train)
end_time = time.time()
elapsed_time3 = end_time - start_time
print("\n-----SVC Model-----\n")
print("education time: ", elapsed_time3)
print("Training data score: ", model3.score(X_train, y_train))

-----SVC Model-----

education time:  129.59862875938416
Training data score:  0.9973213612012375

start_time = time.time()
model4 = make_pipeline(TfidfVectorizer(), RandomForestClassifier())
model4.fit(X_train, y_train)
end_time = time.time()
elapsed_time4 = end_time - start_time
print("\n-----RandomForestClassifier Model-----\n")
print("education time: ", elapsed_time4)
print("Training data score: ", model4.score(X_train, y_train))

-----RandomForestClassifier Model-----

education time:  98.48843431472778
Training data score:  0.999679317890289

# Evaluate the model on the test set
start_time = time.time()
y_pred1 = model1.predict(X_test)
end_time = time.time()
predict_elapsed_time1 = end_time - start_time

```

```

print("\n-----MultinomialNB Model Evaluation on Test
Set-----\n")
print(metrics.classification_report(y_test, y_pred1))
print("Estimation time: ", predict_elapsed_time1)

-----MultinomialNB Model Evaluation on Test Set-----

```

	precision	recall	f1-score	support
anom	0.97	0.98	0.97	4586
norm	0.99	0.98	0.99	8668
accuracy			0.98	13254
macro avg	0.98	0.98	0.98	13254
weighted avg	0.98	0.98	0.98	13254

```

Estimation time: 0.13304519653320312

# Evaluate the model on the test set
start_time = time.time()
y_pred2 = model2.predict(X_test)
end_time = time.time()
predict_elapsed_time2 = end_time - start_time
print("\n-----LogisticRegression Model Evaluation on Test
Set-----\n")
print(metrics.classification_report(y_test, y_pred2))
print("Estimation time: ", predict_elapsed_time2)

-----LogisticRegression Model Evaluation on Test
Set-----

```

	precision	recall	f1-score	support
anom	0.98	0.93	0.96	4586
norm	0.96	0.99	0.98	8668
accuracy			0.97	13254
macro avg	0.97	0.96	0.97	13254
weighted avg	0.97	0.97	0.97	13254

```

Estimation time: 0.08551454544067383

# Evaluate the model on the test set
start_time = time.time()
y_pred3 = model3.predict(X_test)
end_time = time.time()
predict_elapsed_time3 = end_time - start_time
print("\n-----SVC Model Evaluation on Test Set-----\n")

```



```
print(metrics.classification_report(y_test, y_pred3))
print("Estimation time: ", predict_elapsed_time3)
```

-----SVC Model Evaluation on Test Set-----

	precision	recall	f1-score	support
anom	0.99	0.99	0.99	4586
norm	0.99	0.99	0.99	8668
accuracy			0.99	13254
macro avg	0.99	0.99	0.99	13254
weighted avg	0.99	0.99	0.99	13254

Estimation time: 4.390802621841431

Evaluate the model on the test set

```
start_time = time.time()
y_pred4 = model4.predict(X_test)
end_time = time.time()
predict_elapsed_time4 = end_time - start_time
print("\n-----RandomForestClassifier Model Evaluation on Test
Set-----\n")
print(metrics.classification_report(y_test, y_pred4))
print("Estimation time: ", predict_elapsed_time4)
```

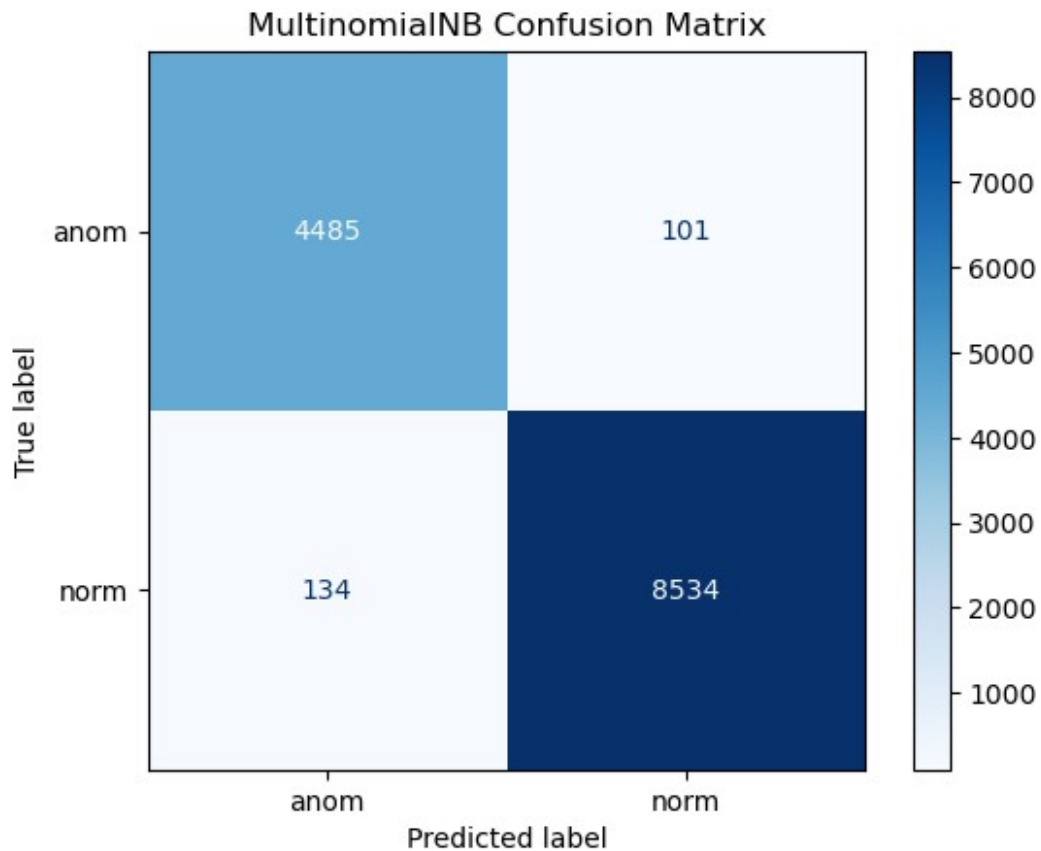
-----RandomForestClassifier Model Evaluation on Test Set-----

	precision	recall	f1-score	support
anom	0.67	1.00	0.80	4586
norm	1.00	0.74	0.85	8668
accuracy			0.83	13254
macro avg	0.83	0.87	0.82	13254
weighted avg	0.88	0.83	0.83	13254

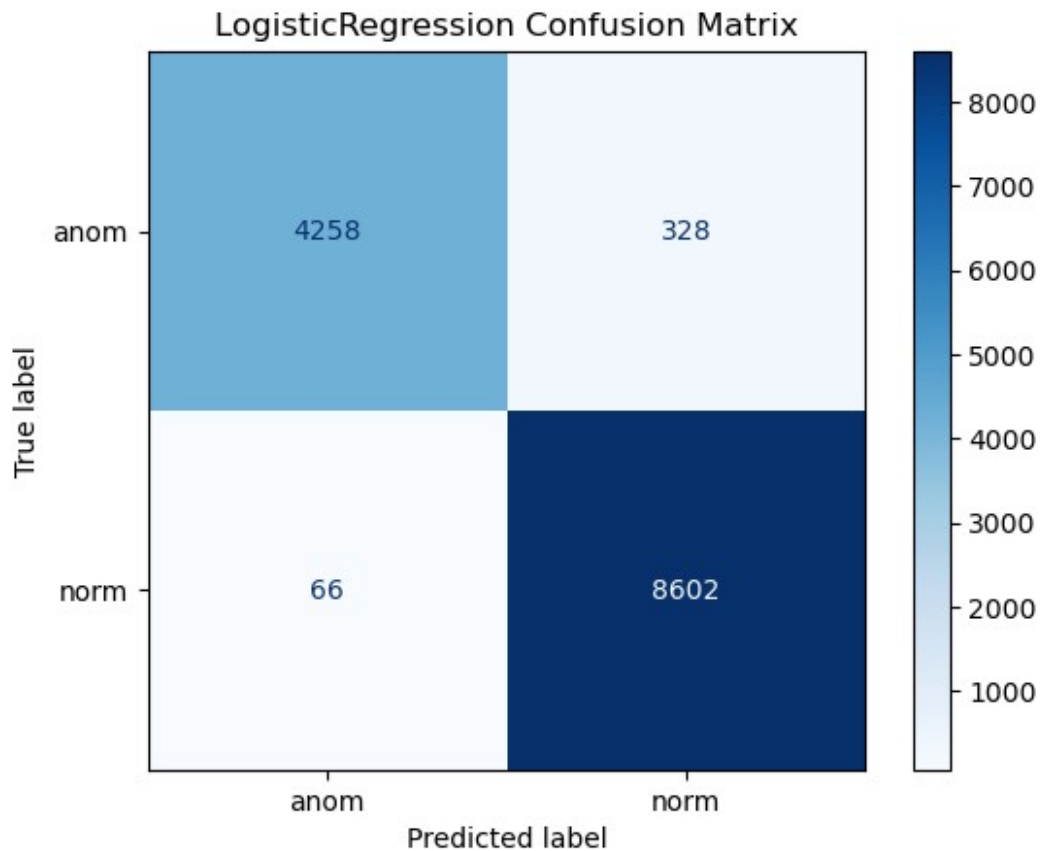
Estimation time: 8.607045650482178

Confusion matrix

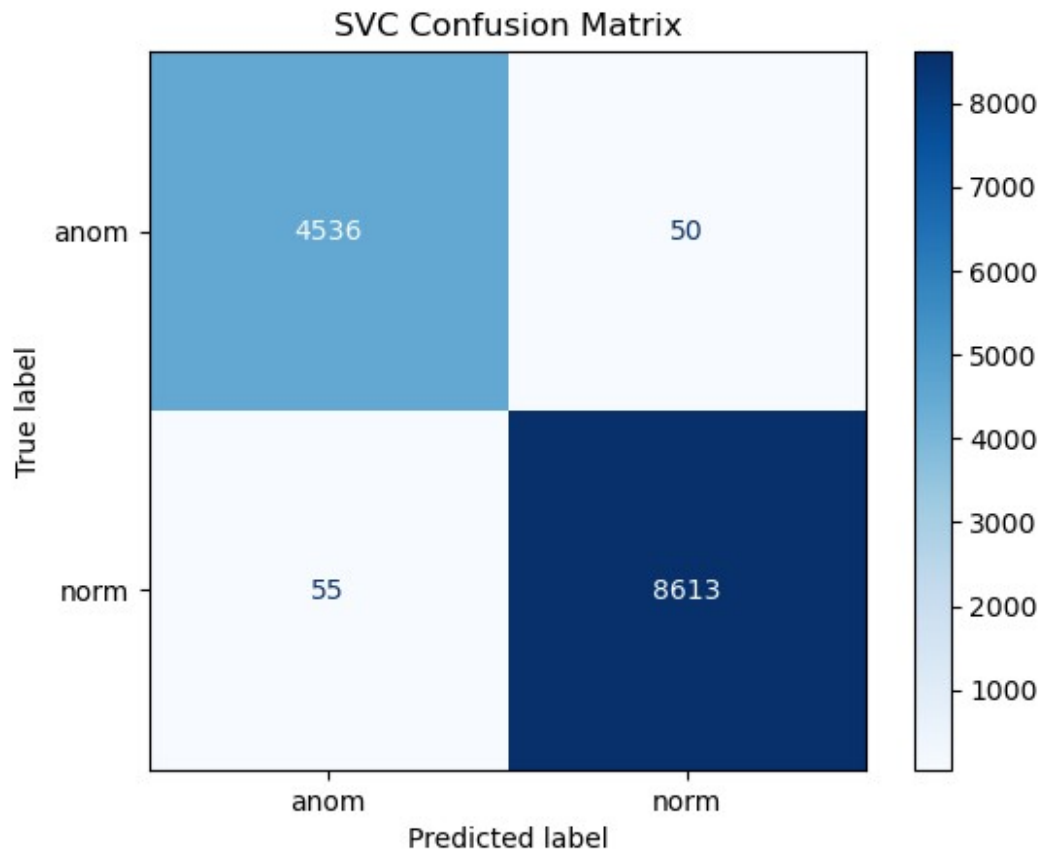
```
cm = confusion_matrix(y_test, y_pred1, labels=model1.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model1.classes_)
disp.plot(cmap='Blues', values_format='d')
plt.title('MultinomialNB Confusion Matrix')
plt.show()
```



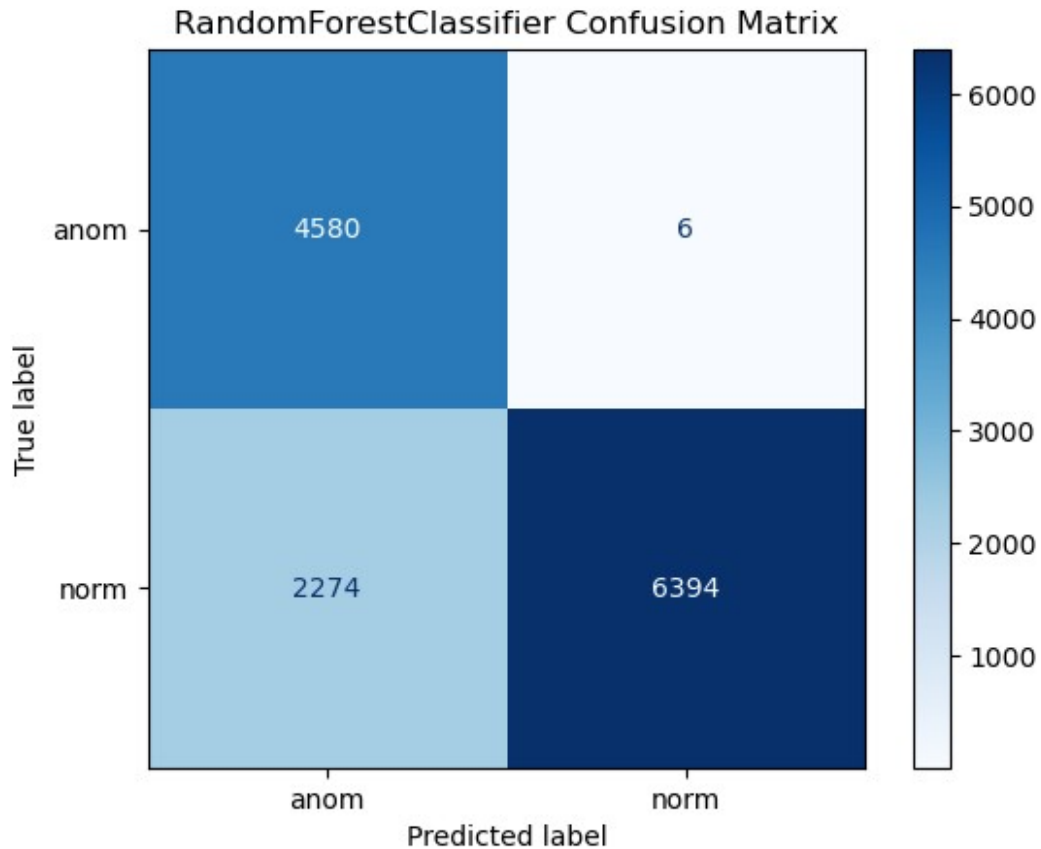
```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred2, labels=model2.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model2.classes_)
disp.plot(cmap='Blues', values_format='d')
plt.title('LogisticRegression Confusion Matrix')
plt.show()
```



```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred3, labels=model3.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model3.classes_)
disp.plot(cmap='Blues', values_format='d')
plt.title('SVC Confusion Matrix')
plt.show()
```

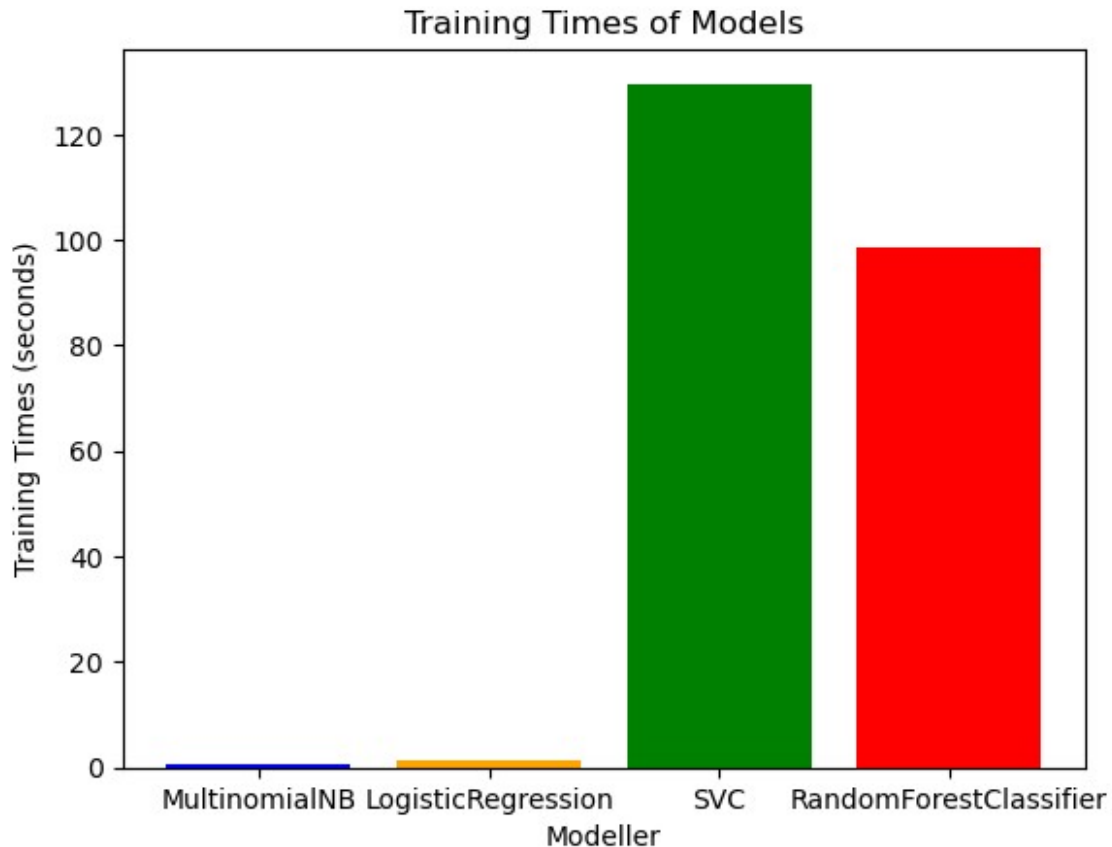


```
# Confusion matrix
cm = confusion_matrix(y_test, y_pred4, labels=model4.classes_)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model4.classes_)
disp.plot(cmap='Blues', values_format='d')
plt.title('RandomForestClassifier Confusion Matrix')
plt.show()
```



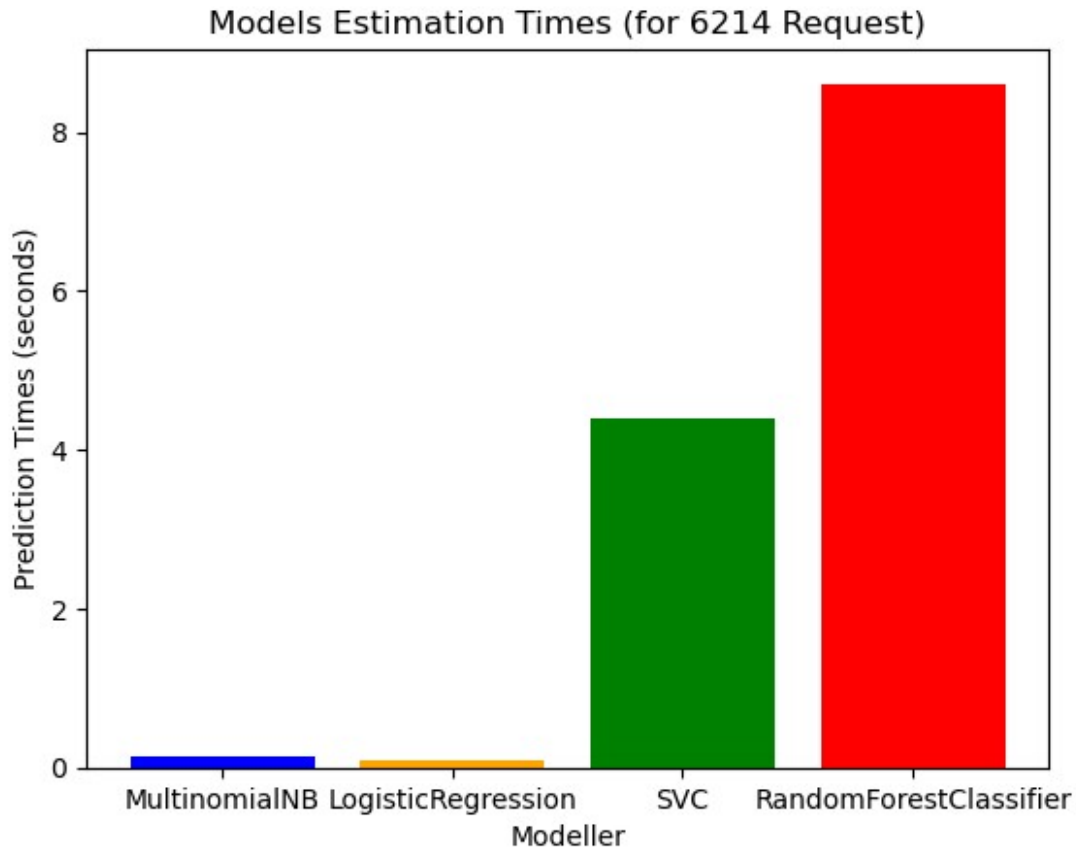
```
# Training times of four models (sample data)
model_names = ["MultinomialNB", "LogisticRegression", "SVC",
               "RandomForestClassifier"]
training_times = [elapsed_time1, elapsed_time2, elapsed_time3,
                  elapsed_time4] # These times should be based on the data you have.

# Visualize training times
plt.bar(model_names, training_times, color=['blue', 'orange', 'green',
      'red'])
plt.xlabel('Modeller')
plt.ylabel('Training Times (seconds)')
plt.title('Training Times of Models')
plt.show()
```



```
# Prediction times of four models (sample data)
model_names = ["MultinomialNB", "LogisticRegression", "SVC",
               "RandomForestClassifier"]
training_times = [predict_elapsed_time1, predict_elapsed_time2,
                  predict_elapsed_time3, predict_elapsed_time4] # These times should be
# based on the data you have.

# Visualize training times
plt.bar(model_names, training_times, color=['blue', 'orange', 'green',
      'red'])
plt.xlabel('Modeller')
plt.ylabel('Prediction Times (seconds)')
plt.title('Models Estimation Times (for 6214 Request)')
plt.show()
```



```
# Endpoint for real-time analysis
@app.route('/analyze_request1', methods=['POST'])
def analyze_request1():
    # Receive incoming request
    client_ip = request.remote_addr
    request_text = request.get_data(as_text=True)

    # Analyze using the model
    prediction = model1.predict([request_text])
    is_malicious = prediction[0] == 'anom'

    # Return warning or information message
    if is_malicious:
        return jsonify({'message': 'Warning: Potential security threat detected!', 'ip': client_ip})
    else:
        return jsonify({'message': 'Normal request: Safe.', 'ip': client_ip})

# Endpoint for real-time analysis
@app.route('/analyze_request2', methods=['POST'])
def analyze_request2():
    # Receive incoming request
```

```

client_ip = request.remote_addr
request_text = request.get_data(as_text=True)

# Analyze using the model
prediction = model2.predict([request_text])
is_malicious = prediction[0] == 'anom'

# Return warning or information message
if is_malicious:
    return jsonify({'message': 'Warning: Potential security threat
detected!', 'ip': client_ip})
else:
    return jsonify({'message': 'Normal request: Safe.', 'ip':
client_ip})

# Endpoint for real-time analysis
@app.route('/analyze_request3', methods=['POST'])
def analyze_request3():
    # Gelen isteği al
    client_ip = request.remote_addr
    request_text = request.get_data(as_text=True)

    # Analyze using the model
    prediction = model3.predict([request_text])
    is_malicious = prediction[0] == 'anom'

    # Return warning or information message
    if is_malicious:
        return jsonify({'message': 'Warning: Potential security threat
detected!', 'ip': client_ip})
    else:
        return jsonify({'message': 'Normal request: Safe.', 'ip':
client_ip})

# Endpoint for real-time analysis
@app.route('/analyze_request4', methods=['POST'])
def analyze_request4():
    # Receive incoming request
    client_ip = request.remote_addr
    request_text = request.get_data(as_text=True)

    # Analyze using the model
    prediction = model4.predict([request_text])
    is_malicious = prediction[0] == 'anom'

    # Return warning or information message
    if is_malicious:
        return jsonify({'message': 'Warning: Potential security threat
detected!', 'ip': client_ip})
    else:

```



```
        return jsonify({'message': 'Normal request: Safe.', 'ip':  
client_ip})
```

```
# Starting the web server
```

```
if __name__ == '__main__':  
    app.run(port=8080)
```

```
* Serving Flask app '__main__'  
* Debug mode: off
```

WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.

```
* Running on http://127.0.0.1:8080
```

Press CTRL+C to quit

```
127.0.0.1 - - [01/Feb/2024 12:32:57] "GET / HTTP/1.1" 404 -
```

```
127.0.0.1 - - [01/Feb/2024 12:32:57] "GET /favicon.ico HTTP/1.1" 404 -
```

```
### curl commands for testing
```

```
# curl -X POST http://localhost:5000/analyze_request1 -d "SELECT *  
FROM users WHERE id = 1"
```

```
# curl -X POST http://localhost:5000/analyze_request1 -d "SELECT *  
FROM users WHERE id = 1 OR 1=1"
```

```
# curl -X POST http://localhost:5000/analyze_request1 -d "SELECT *  
FROM users WHERE id = 1; DROP TABLE users"
```

```
# curl -X POST http://localhost:5000/analyze_request1 -d  
"hasan@hotmail.com"
```