```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn import metrics
import time
```

```
# reading the file using pandas library

df=pd.read_csv(r"/spam.csv")
```

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

```
# Exploring the dataset
df.head(5)
```

	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

```
df.tail()
```

```
# lets check the features
df.columns
```

```
df.describe()
```

```
# check for the datatypes
df.dtypes
```

```
df.info()
```

```
# checking for null values
df.isna().sum()
```

```
df.size
```

```
df.size
```

```
df.head(10)
```

```
# (we have to create a new column to make a difference between spam and not sp
```

```
df['spam']=df['Category'].apply(lambda x:1 if x=='spam' else 0)
```

```
df
```

```
df.groupby('Category').count()
```

```
X_train,X_test,y_train,y_test=train_test_split(df.Message,df.spam,test_size=0.
```

```
# lets check the size of  our data
```

```
X_train.size
```

```
X_test.size
```

```
y_train.size
```

```
y_test.size
```

we convert the textual data into a numerical form that machine learning models can understand.

```
vectorizer=TfidfVectorizer()
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized= vectorizer.transform(X_test)
```

```
# naive bayes
```

```
nb_model = MultinomialNB()
start_time = time.time()
nb_model.fit(X_train_vectorized, y_train)
nb_time = time.time() - start_time
nb_predictions = nb_model.predict(X_test_vectorized)
```

```
# Decision Tree (J48 equivalent)
dt_model = DecisionTreeClassifier()
start_time = time.time()
dt_model.fit(X_train_vectorized, y_train)
dt_time = time.time() - start_time
dt_predictions = dt_model.predict(X_test_vectorized)
```

Evaluate Performance of the models

```
def evaluate_model(predictions, model_name):
   accuracy = metrics.accuracy_score(y_test, predictions)
   error_rate = 1 - accuracy
   print(f"{model_name} Accuracy: {accuracy:.4f}")
   print(f"{model_name} Error Rate: {error_rate:.4f}")

# Evaluate Naive Bayes
evaluate_model(nb_predictions, "Naive Bayes")
print(f"Naive Bayes Processing Time: {nb_time:.4f} seconds\n")

# Evaluate Decision Tree
evaluate_model(dt_predictions, "Decision Tree (J48)")
print(f"Decision Tree Processing Time: {dt_time:.4f} seconds")
```