A1: Spam Filter Using Naive Bayes Algorithm

```
#import required packages
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import CountVectorizer
```

```
#import dataset
spam_df = pd.read_csv('spam.csv')
```

```
#lets look our data
spam_df.head()
```

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

```
#inspect the data
spam_df.groupby('Category').describe()
```

	Message			
	count	unique	top	freq
Category				
ham	4825	4516	Sorry, I'll call later	30
spam	747	641	Please call our customer service representativ...	4

```
#convert spam/ham into numerical data, creating new column called 'spam'
spam_df['spam'] = spam_df['Category'].apply(lambda x: 1 if x == 'spam' else 0)
```

```
#create train test split
x_train, x_test, y_train, y_test = train_test_split(spam_df.Message, spam_df.spam, test_size=0.25)
```

```
#find word coun and store data as matrix
cv = CountVectorizer()
x_train_count = cv.fit_transform(x_train.values)
```

```
#train model
model = MultinomialNB()
model.fit(x_train_count, y_train)
```

```
▾ MultinomialNB
MultinomialNB()
```

```
#pre-test ham
email_ham = ['baseball ticket later']
email_ham_count = cv.transform(email_ham)
model.predict(email_ham_count)
```

```
array([0])
```

```
#pre test spam
email_spam = ['reward money click']
email_spam_count = cv.transform(email_spam)
model.predict(email_spam_count)
```

```
array([1])
```

```
#test_model
x_test_count = cv.transform(x_test)
model.score(x_test_count, y_test)
```

```
0.9863603732950467
```

A3: Split sample data into training & testing sets.

```
#import required libraries
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
#read the dataset
datasets = pd.read_csv('DataSplit.csv')
```

```
#check the data set using head() function
datasets.head()
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3

iloc[] ▶ Purely integer-location based indexing for selection by position.

```
#get the location
x = datasets.iloc[:, :-1]
y = datasets.iloc[:, :-1]
```

```
#split tha datasets using train_test_split fucntion
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.05, random_state=0)
```

```
#check the size of train dataset
xtrain = x_train.shape
ytrain = y_train.shape
print(f'Dataset size of x_test: {xtrain}')
print(f'Dataset size of y_test: {ytrain}')
```

```
Dataset size of x_test: (393, 7)
Dataset size of y_test: (393, 7)
```

```
#check the test datasets shape
xtest = x_test.shape
ytest = y_test.shape
print(f'Dataset size of x_test: {xtest}')
print(f'Dataset size of y_test: {ytest}')
```

```
Dataset size of x_test: (21, 7)
Dataset size of y_test: (21, 7)
```

```
#check the _x_train dataset
x_train
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude

```
#check the y_train dataset
y_train
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
37	38	2013.167	12.0	1360.13900	1	24.95204	121.54842
334	335	2012.917	30.0	1013.34100	5	24.99006	121.53460
54	55	2013.083	16.1	289.32480	5	24.98203	121.54348
145	146	2012.917	2.1	451.24380	5	24.97563	121.54694
284	285	2012.917	15.0	383.28050	7	24.96735	121.54464
...	...	...	...	...	...	...	...
323	324	2013.417	28.6	197.13380	6	24.97631	121.54436
192	193	2013.167	43.8	57.58945	7	24.96750	121.54069
117	118	2013.000	13.6	4197.34900	0	24.93885	121.50383
47	48	2013.583	35.9	640.73910	3	24.97563	121.53715
172	173	2013.583	6.6	90.45606	9	24.97433	121.54310

```
#check the x_test dataset
x_test
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
356	357	2012.833	10.3	211.44730	1	24.97417	121.52999
170	171	2013.333	24.0	4527.68700	0	24.94741	121.49628
224	225	2013.333	34.5	324.94190	6	24.97814	121.54170
331	332	2013.333	25.6	4519.69000	0	24.94826	121.49587
306	307	2013.500	14.4	169.98030	1	24.97369	121.52979
325	326	2013.083	36.6	488.81930	8	24.97015	121.54494
150	151	2013.250	35.8	170.73110	7	24.96719	121.54269
10	11	2013.083	34.8	405.21340	1	24.97349	121.53372
21	22	2013.417	10.5	279.17260	7	24.97528	121.54541
268	269	2013.417	17.2	390.56840	5	24.97937	121.54245
316	317	2013.250	13.3	250.63100	7	24.96606	121.54297
59	60	2013.083	13.3	336.05320	5	24.95776	121.53438
402	403	2012.833	12.7	187.48230	1	24.97388	121.52981
198	199	2013.083	34.0	157.60520	7	24.96628	121.54196
348	349	2012.833	4.6	259.66070	6	24.97585	121.54516
76	77	2013.583	35.9	616.40040	3	24.97723	121.53767
264	265	2013.167	32.6	493.65700	7	24.96968	121.54522
164	165	2012.833	0.0	185.42960	0	24.97110	121.53170
12	13	2012.917	13.0	492.23130	5	24.96515	121.53737
188	189	2012.917	34.8	190.03920	8	24.97707	121.54312

```
#check the y_test Datasets
y_test
```

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude
356	357	2012.833	10.3	211.44730	1	24.97417	121.52999
170	171	2013.333	24.0	4527.68700	0	24.94741	121.49628
224	225	2013.333	34.5	324.94190	6	24.97814	121.54170
331	332	2013.333	25.6	4519.69000	0	24.94826	121.49587
306	307	2013.500	14.4	169.98030	1	24.97369	121.52979
325	326	2013.083	36.6	488.81930	8	24.97015	121.54494
150	151	2013.250	35.8	170.73110	7	24.96719	121.54269
10	11	2013.083	34.8	405.21340	1	24.97349	121.53372
21	22	2013.417	10.5	279.17260	7	24.97528	121.54541
268	269	2013.417	17.2	390.56840	5	24.97937	121.54245
316	317	2013.250	13.3	250.63100	7	24.96606	121.54297
59	60	2013.083	13.3	336.05320	5	24.95776	121.53438
402	403	2012.833	12.7	187.48230	1	24.97388	121.52981
198	199	2013.083	34.0	157.60520	7	24.96628	121.54196
348	349	2012.833	4.6	259.66070	6	24.97585	121.54516
264	265	2013.167	32.6	493.65700	7	24.96968	121.54522
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

A4: Perform Feature Engineering Operation on Raw Data

```
#import dependencies
import numpy as np
import pandas as pd
```

```
#create dataframes
data={
  'candy variety':['chocolate hearts','sour jelly','candy canes','sour jelly','fruit drops'],
    'Date and Time':['09-02-2020 14:05','24-10-2020 18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
    'Day':['sunday','saturday','friday','sunday','sunday'],
    'Length':[3,3.5,3.5,3.5,3],
    'Breadth':[2,2,2.5,2,3],
    'Price':[7.5,7.6,8,7.6,9]
  }
df=pd.DataFrame(data)
df.head()
```

	candy variety	Date and Time	Day	Length	Breadth	Price
0	chocolate hearts	09-02-2020 14:05	sunday	3.0	2.0	7.5
1	sour jelly	24-10-2020 18:00	saturday	3.5	2.0	7.6
2	candy canes	18-12-2020 20:13	friday	3.5	2.5	8.0
3	sour jelly	25-10-2020 10:00	sunday	3.5	2.0	7.6
4	fruit drops	18-10-2020 15:46	sunday	3.0	3.0	9.0

```
#change the format of 'Date and Time'
df['Date and Time']=pd.to_datetime(df['Date and Time'],format="%d-%m-%Y %H:%M")
print(df)
```

```
      candy variety       Date and Time       Day  Length  Breadth  Price
0  chocolate hearts 2020-02-09 14:05:00    sunday     3.0      2.0    7.5
1        sour jelly 2020-10-24 18:00:00  saturday     3.5      2.0    7.6
2       candy canes 2020-12-18 20:13:00    friday     3.5      2.5    8.0
3        sour jelly 2020-10-25 10:00:00    sunday     3.5      2.0    7.6
4       fruit drops 2020-10-18 15:46:00    sunday     3.0      3.0    9.0
```

```
# creating new feature Date from existing feature Date and Time #
df['Date']=df['Date and Time'].dt.date
print(df[['candy variety','Date']])
```

```
      candy variety        Date
0  chocolate hearts  2020-02-09
1        sour jelly  2020-10-24
2       candy canes  2020-12-18
3        sour jelly  2020-10-25
4       fruit drops  2020-10-18
```

```
# creating weekend from days
df['weekend']=np.where(df['Day'].isin(['saturday','sunday']),1,0)
print(df[['candy variety','Date','weekend']])
```

```
      candy variety        Date  weekend
0  chocolate hearts  2020-02-09        1
1        sour jelly  2020-10-24        1
2       candy canes  2020-12-18        0
3        sour jelly  2020-10-25        1
4       fruit drops  2020-10-18        1
```

```
#create a new data set
data={
  'candy variety':['chocolate hearts','sour jelly','candy canes','sour jelly','fruit drops'],
    'Date and Time':['09-02-2020 14:05','24-10-2020 18:00','18-12-2020 20:13','25-10-2020 10:00','18-10-2020 15:46'],
    'Day':['sunday','saturday','friday','sunday','sunday'],
    'Length':[3,3.5,3.5,3.5,3],
    'Breadth':[2,2,2.5,2,3],
    'Price':[7.5,7.6,8,7.6,9]
  }
df=pd.DataFrame(data)
df.head()
```

	candy variety	Date and Time	Day	Length	Breadth	Price
0	chocolate hearts	09-02-2020 14:05	sunday	3.0	2.0	7.5
1	sour jelly	24-10-2020 18:00	saturday	3.5	2.0	7.6
2	candy canes	18-12-2020 20:13	friday	3.5	2.5	8.0
3	sour jelly	25-10-2020 10:00	sunday	3.5	2.0	7.6

```
#Appending row with missing values
df['Date and Time']=pd.to_datetime(df['Date and Time'],format="%d-%m-%Y %H:%M")
df.loc[len(df.index)]=[np.NaN,'22-10-2020 17:24','thursday',3.5,2,np.NaN]
print(df)
```

```
      candy variety        Date and Time       Day  Length  Breadth  Price
0  chocolate hearts  2020-02-09 14:05:00    sunday     3.0      2.0    7.5
1        sour jelly  2020-10-24 18:00:00  saturday     3.5      2.0    7.6
2       candy canes  2020-12-18 20:13:00    friday     3.5      2.5    8.0
3        sour jelly  2020-10-25 10:00:00    sunday     3.5      2.0    7.6
4       fruit drops  2020-10-18 15:46:00    sunday     3.0      3.0    9.0
5               NaN     22-10-2020 17:24  thursday     3.5      2.0    NaN
```

```
# Imputation
df['candy variety']=df['candy variety'].fillna(df['candy variety'].mode()[0])
df['Price']=df['Price'].fillna(df['Price'].mean())
print(df)
```

```
      candy variety        Date and Time       Day  Length  Breadth  Price
0  chocolate hearts  2020-02-09 14:05:00    sunday     3.0      2.0   7.50
1        sour jelly  2020-10-24 18:00:00  saturday     3.5      2.0   7.60
2       candy canes  2020-12-18 20:13:00    friday     3.5      2.5   8.00
3        sour jelly  2020-10-25 10:00:00    sunday     3.5      2.0   7.60
4       fruit drops  2020-10-18 15:46:00    sunday     3.0      3.0   9.00
5        sour jelly     22-10-2020 17:24  thursday     3.5      2.0   7.94
```

```
# Discretization
df['Type of Day']=np.where(df['Day'].isin(['saturday','sunday']),'weekend','weekday')
df[['candy variety','Day','Type of Day']]
print(df)
```

```
      candy variety        Date and Time       Day  Length  Breadth  Price  \
0  chocolate hearts  2020-02-09 14:05:00    sunday     3.0      2.0   7.50
1        sour jelly  2020-10-24 18:00:00  saturday     3.5      2.0   7.60
2       candy canes  2020-12-18 20:13:00    friday     3.5      2.5   8.00
3        sour jelly  2020-10-25 10:00:00    sunday     3.5      2.0   7.60
4       fruit drops  2020-10-18 15:46:00    sunday     3.0      3.0   9.00
5        sour jelly     22-10-2020 17:24  thursday     3.5      2.0   7.94

  Type of Day
0     weekend
1     weekend
2     weekday
3     weekend
4     weekend
5     weekday
```

```
#Categorical Encoding
for x in df['Type of Day'].unique():df[x]=np.where(df['Type of Day']==x,1,0)
print(df[['candy variety','Day','Type of Day','weekend','weekday']])
```

```
      candy variety       Day Type of Day  weekend  weekday
0  chocolate hearts    sunday     weekend        1        0
1        sour jelly  saturday     weekend        1        0
2       candy canes    friday     weekday        0        1
3        sour jelly    sunday     weekend        1        0
4       fruit drops    sunday     weekend        1        0
5        sour jelly  thursday     weekday        0        1
```

```
# Feature Splitting
df['Date and Time']=pd.to_datetime(df['Date and Time'])
df['Date']=df['Date and Time'].dt.date
print(df[['candy variety','Date']])
```

```
      candy variety        Date
0  chocolate hearts  2020-02-09
1        sour jelly  2020-10-24
2       candy canes  2020-12-18
3        sour jelly  2020-10-25
4       fruit drops  2020-10-18
5        sour jelly  2020-10-22
```