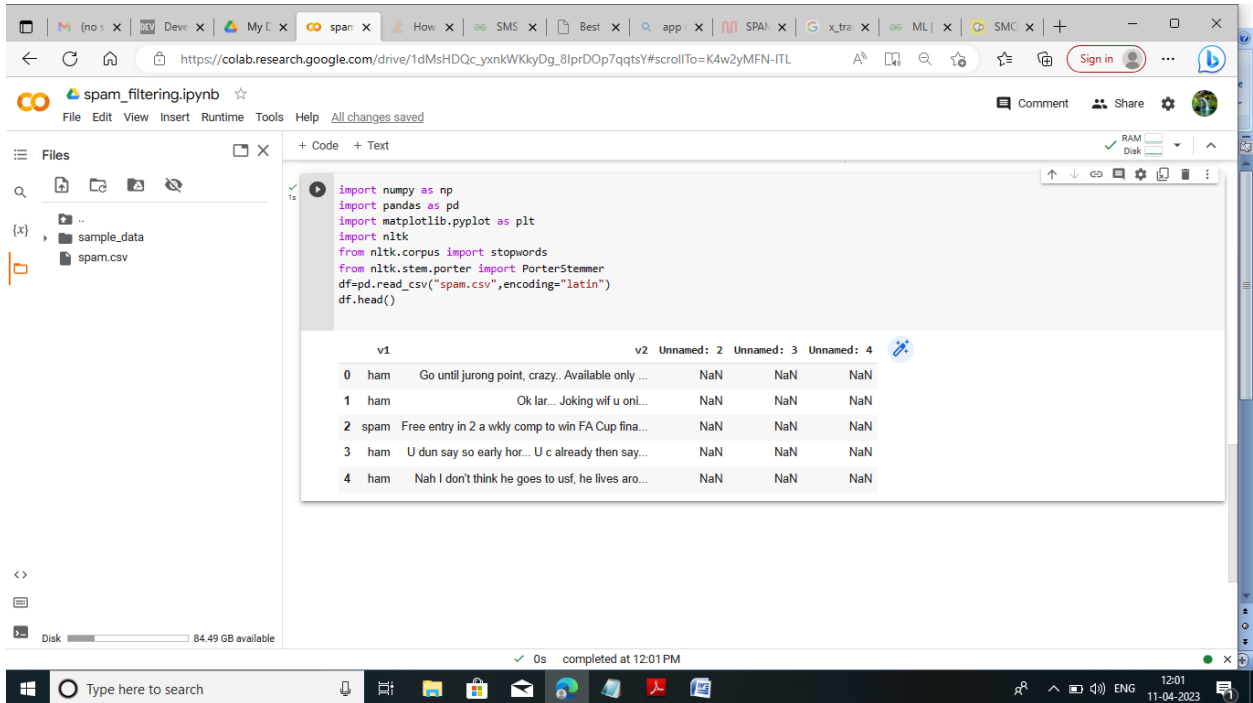


Importing the libraries and Read the Dataset



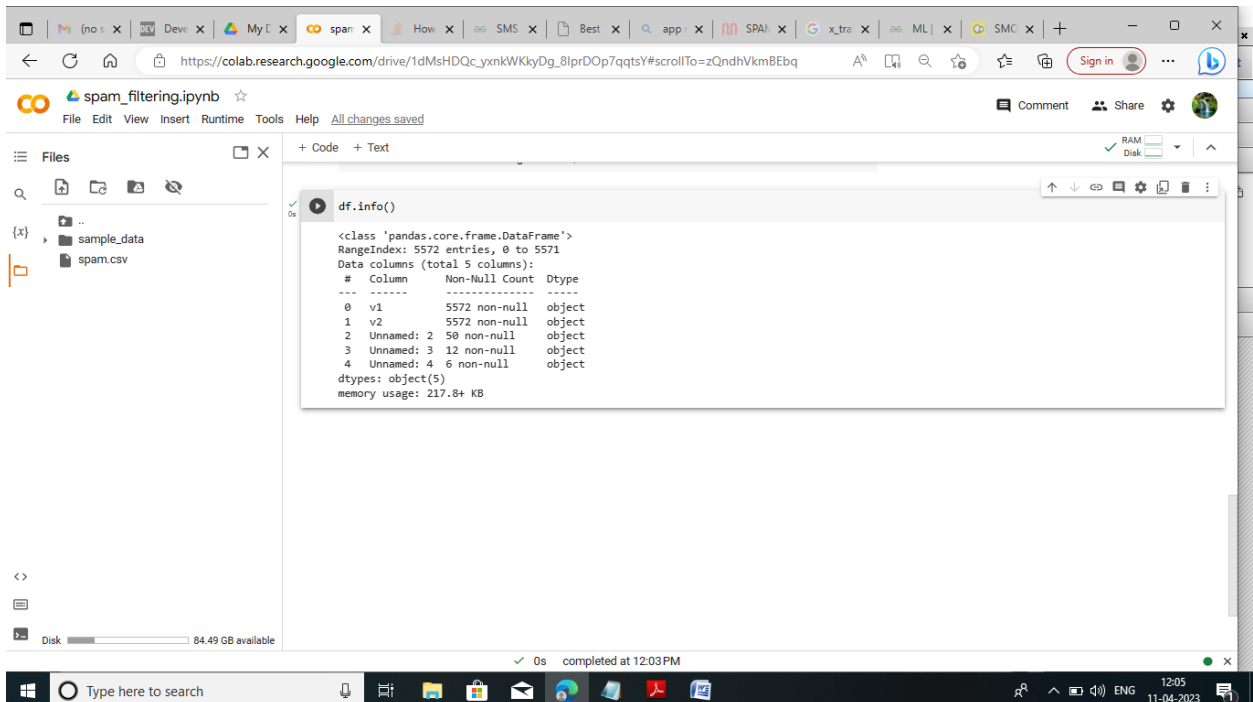
The screenshot shows a Google Colab notebook titled "spam_filtering.ipynb". The code cell contains the following Python code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
df=pd.read_csv("spam.csv",encoding="latin")
df.head()
```

The output of the code is a preview of the first five rows of the dataset:

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Handling missing values



The screenshot shows the same Google Colab notebook with the code cell containing:

```
df.info()
```

The output of the code is the following information:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  -
 0   v1            5572 non-null   object  
 1   v2            5572 non-null   object  
 2   Unnamed: 2    50 non-null     object  
 3   Unnamed: 3    12 non-null     object  
 4   Unnamed: 4    6 non-null      object  
dtypes: object(5)
memory usage: 217.8+ KB
```

Colab interface showing a Jupyter Notebook titled "spam_filtering.ipynb". The code cell contains:

```
df.isna().sum()
```

The output shows the count of missing values for each column:

```
v1      0
v2      0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

The file explorer on the left shows a folder named "sample_data" containing a file named "spam.csv". The status bar indicates 84.49 GB available and the notebook completed at 12:06 PM.

Colab interface showing the same Jupyter Notebook "spam_filtering.ipynb". The code cell contains:

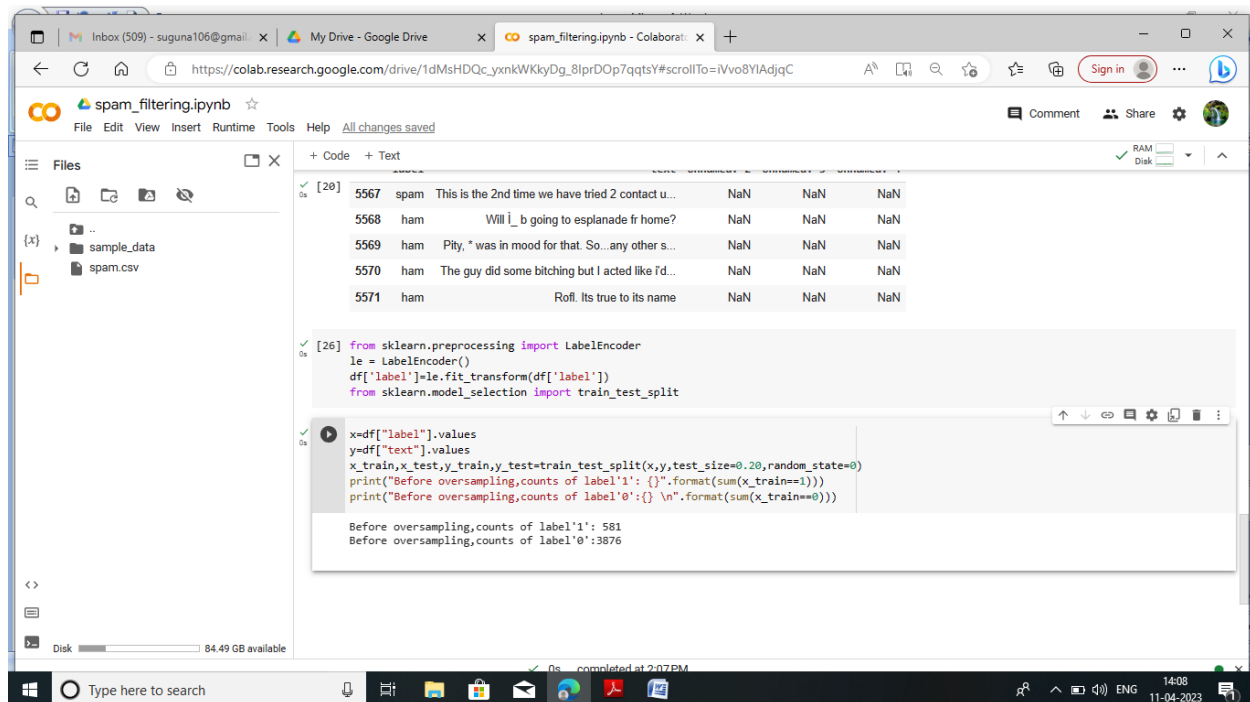
```
df.rename({"v1": "label", "v2": "text"}, inplace=True, axis=1)
df.head()
df.tail()
```

The output displays the first and last rows of the DataFrame:

	label	text	Unnamed: 2	Unnamed: 3	Unnamed: 4
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will I_b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, *was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like I'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

The status bar indicates the notebook completed at 12:21 PM.

Handling Categorical Values



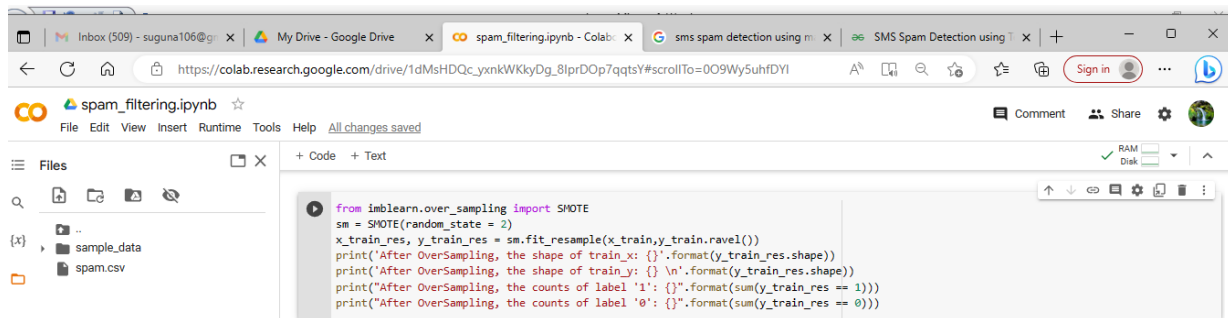
The screenshot shows a Google Colab notebook titled "spam_filtering.ipynb". The left sidebar displays a file explorer with "sample_data" and "spam.csv". The main area shows a table of data with columns for index, label, text, and three numerical columns (all containing NaN). Below the table, code cells are visible. The first code cell imports LabelEncoder and train_test_split. The second code cell defines x and y, and splits the data. The output of the second cell shows the counts of labels before oversampling: 581 for label '1' and 3876 for label '0'.

5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will i_b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, ' was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Roff. Its true to its name	NaN	NaN	NaN

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['label']=le.fit_transform(df['label'])
from sklearn.model_selection import train_test_split

x=df['label'].values
y=df['text'].values
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
print("Before oversampling,counts of label'1': {}".format(sum(x_train==1)))
print("Before oversampling,counts of label'0':{} \n".format(sum(x_train==0)))

Before oversampling,counts of label'1': 581
Before oversampling,counts of label'0':3876
```



The screenshot shows the same Google Colab notebook, but with a new code cell. This cell imports SMOTE from imblearn.over_sampling and applies it to the training data. The code prints the shape of the training data before and after oversampling, and the counts of labels '1' and '0' before and after oversampling.

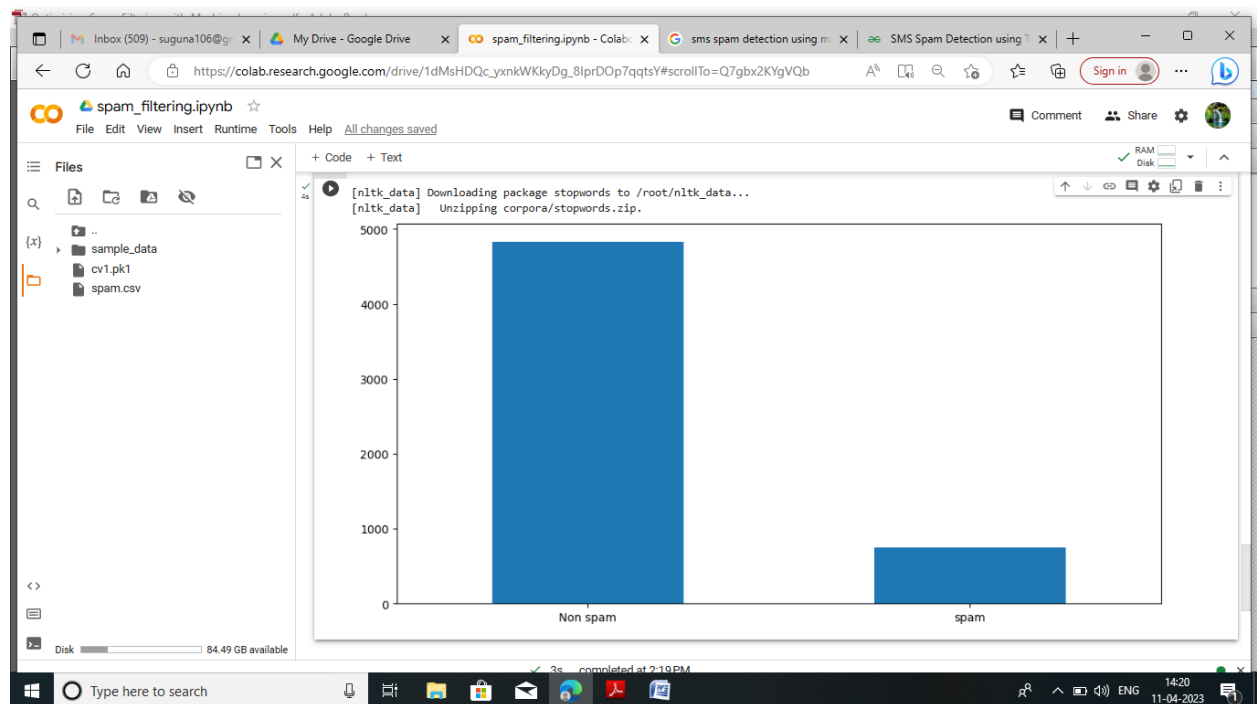
```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state = 2)
x_train_res, y_train_res = sm.fit_resample(x_train,y_train.ravel())
print('After OverSampling, the shape of train_x: {}'.format(y_train_res.shape))
print('After OverSampling, the shape of train_y: {} \n'.format(y_train_res.shape))
print("After OverSampling, the counts of label '1': {}".format(sum(y_train_res == 1)))
print("After OverSampling, the counts of label '0': {}".format(sum(y_train_res == 0)))
```

Before OverSampling, counts of label '1': 581
Before OverSampling, counts of label '0': 3876

After OverSampling, the shape of train_x: (7752, 7163)
After OverSampling, the shape of train_y: (7752,)

After OverSampling, counts of label '1': 3876
After OverSampling, counts of label '0': 3876

Visual analysis



Testing the model

```
y_pred=model.predict(X_test)
y_pred

35/35 [=====] - 2s 29ms/step
array([[1.5844109e-15],
       [4.4117199e-04],
       [1.1517070e-18],
       ...,
       [2.0661259e-08],
       [3.8018154e-17],
       [1.2099350e-12]], dtype=float32)

y_pr = np.where(y_pred>0.5,1,0)

y_test
array([0, 0, 0, ..., 0, 0, 0])

from sklearn.metrics import confusion_matrix,accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test,y_pr)
print(cm)
print('Accuracy Score Is:- ',score*100)

[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816
```

Compare the model

```
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test, y_pred)
print(cm)
print('Accuracy Score Is Naive Bayes:- ', score*100)
```

```
[[935  14]
 [ 13 153]]
Accuracy Score Is:- 97.57847533632287
```

```
cm = confusion_matrix(y_test, y_pred)
score = accuracy_score(y_test, y_pred)
print(cm)
print('Accuracy Score Is:- ', score*100)

cm1 = confusion_matrix(y_test, y_pred1)
score1 = accuracy_score(y_test, y_pred1)
print(cm1)
print('Accuracy Score Is:- ', score1*100)
```

```
[[796 153]
 [ 17 149]]
Accuracy Score Is:- 84.75336322869956
[[855  94]
 [ 14 152]]
Accuracy Score Is:- 90.31390134529148

121/121 [=====] - 24s 196ms/step - loss: 0.0120 - accuracy: 0.9974
Epoch 9/10
121/121 [=====] - 23s 193ms/step - loss: 0.0103 - accuracy: 0.9977
Epoch 10/10
111/121 [=====>...] - ETA: 1s - loss: 0.0128 - accuracy: 0.9972WARNING:tensorflow:

```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test, y_pr)
print(cm)
print('Accuracy Score Is:- ', score*100)
```

```
[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pr)
score = accuracy_score(y_test, y_pr)
print(cm)
print('Accuracy Score Is:- ', score*100)
```

```
[[937  12]
 [ 16 150]]
Accuracy Score Is:- 97.48878923766816
```