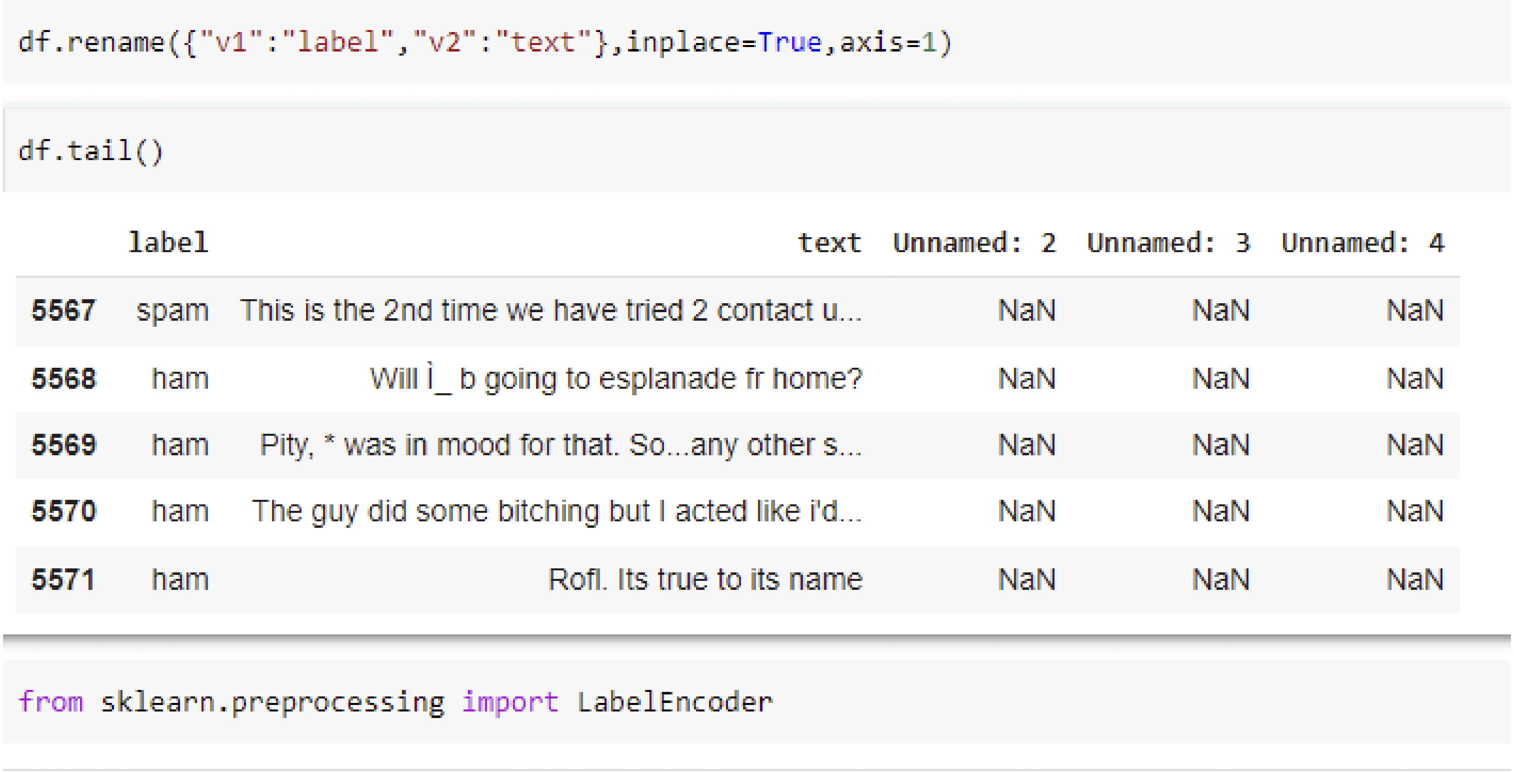
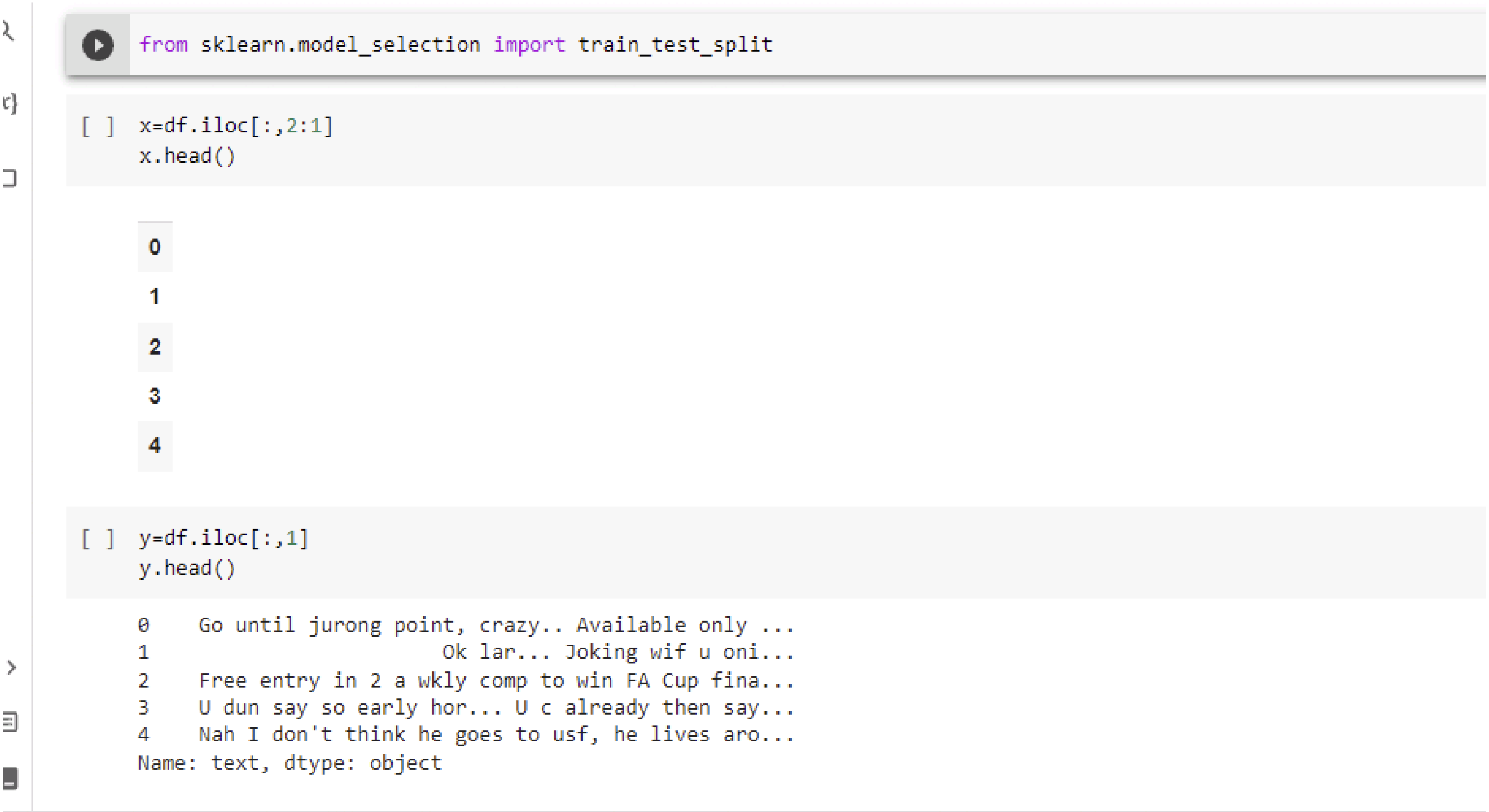


```
df.rename({"v1":"label","v2":"text"},inplace=True,axis=1)
```

```
df.tail()
```

	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 Ì_ 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

```
from sklearn.preprocessing import LabelEncoder
```



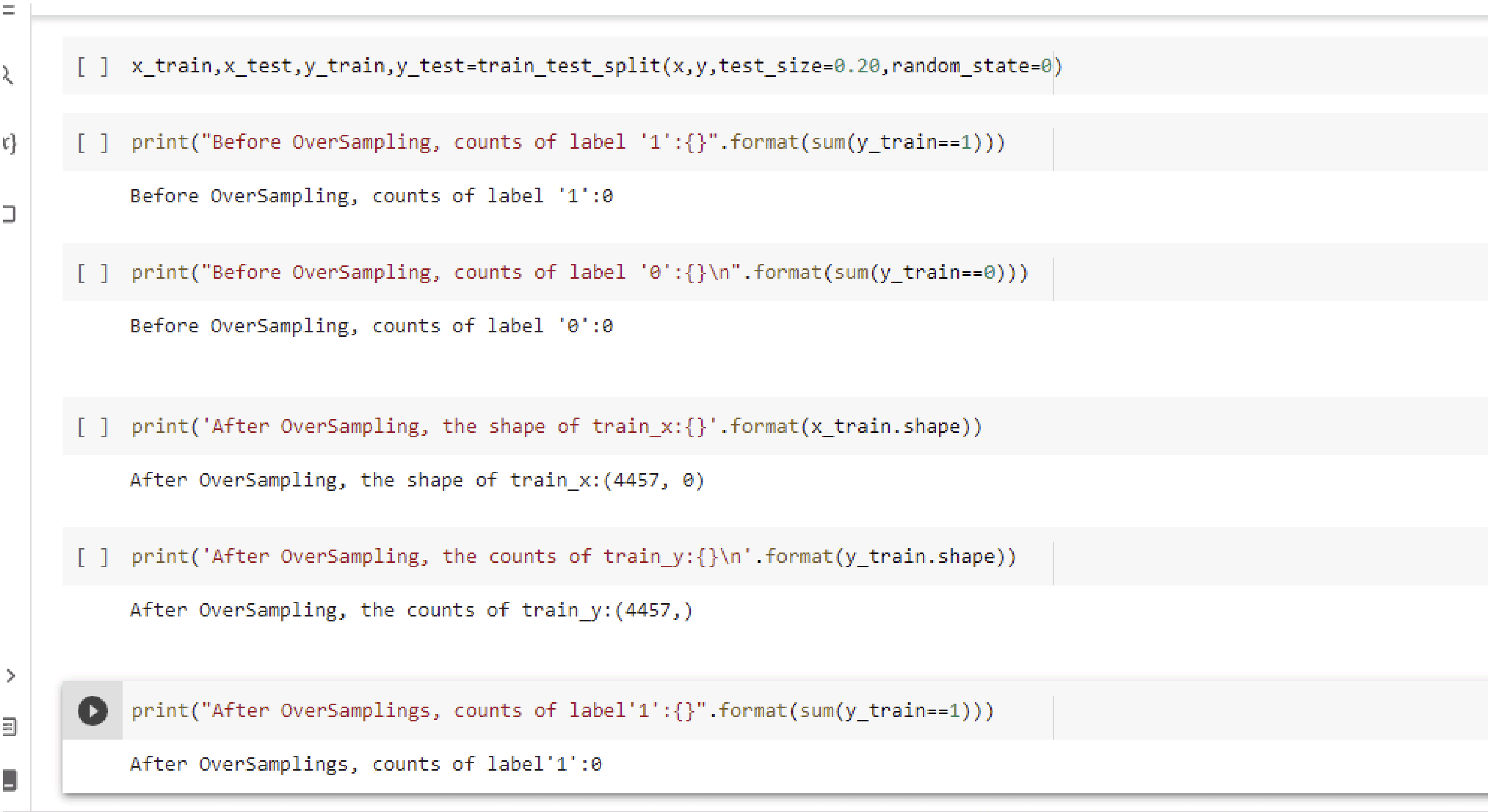
```
from sklearn.model_selection import train_test_split
```

```
[ ]  x=df.iloc[:,2:1]
     x.head()
```

```
0
1
2
3
4
```

```
[ ]  y=df.iloc[:,1]
     y.head()
```

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



```
[ ]  x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

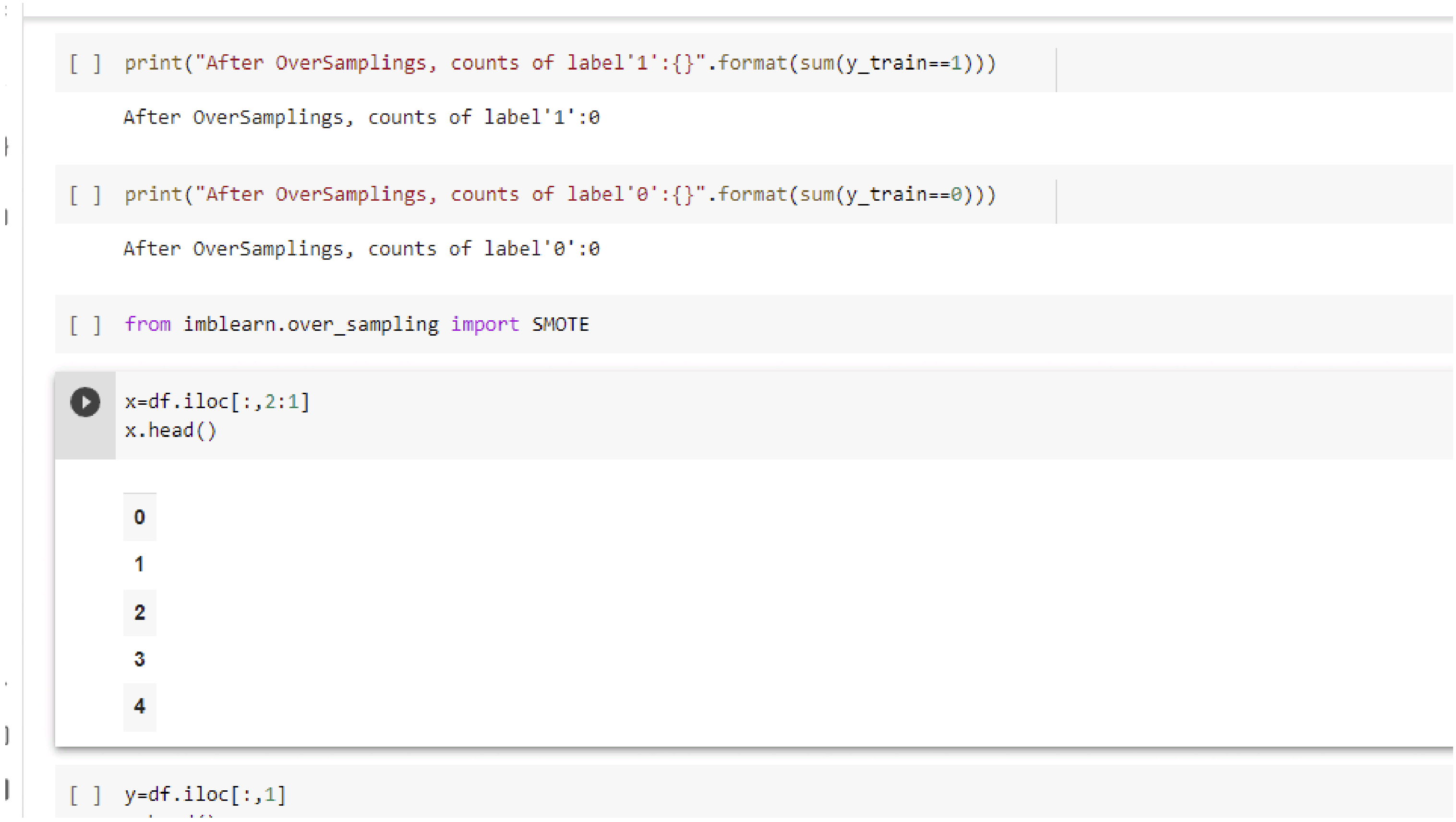
```
[ ]  print("Before OverSampling, counts of label '1':{}".format(sum(y_train==1)))

     Before OverSampling, counts of label '1':0
```

```
[ ]  print("Before OverSampling, counts of label '0':{}\n".format(sum(y_train==0)))

     Before OverSampling, counts of label '0':0
```

```
[ ]  print('After OverSampling, the shape of train_x:{}'.format(x_train.shape))

     After OverSampling, the shape of train_x:(4457, 0)
```

```
[ ]  print('After OverSampling, the counts of train_y:{}\n'.format(y_train.shape))

     After OverSampling, the counts of train_y:(4457,)
```

```
     print("After OverSamplings, counts of label'1':{}".format(sum(y_train==1)))

     After OverSamplings, counts of label'1':0
```



```
print("After OverSamplings, counts of label'1':{}".format(sum(y_train==1)))
```

```
After OverSamplings, counts of label'1':0
```

```
print("After OverSamplings, counts of label'0':{}".format(sum(y_train==0)))
```

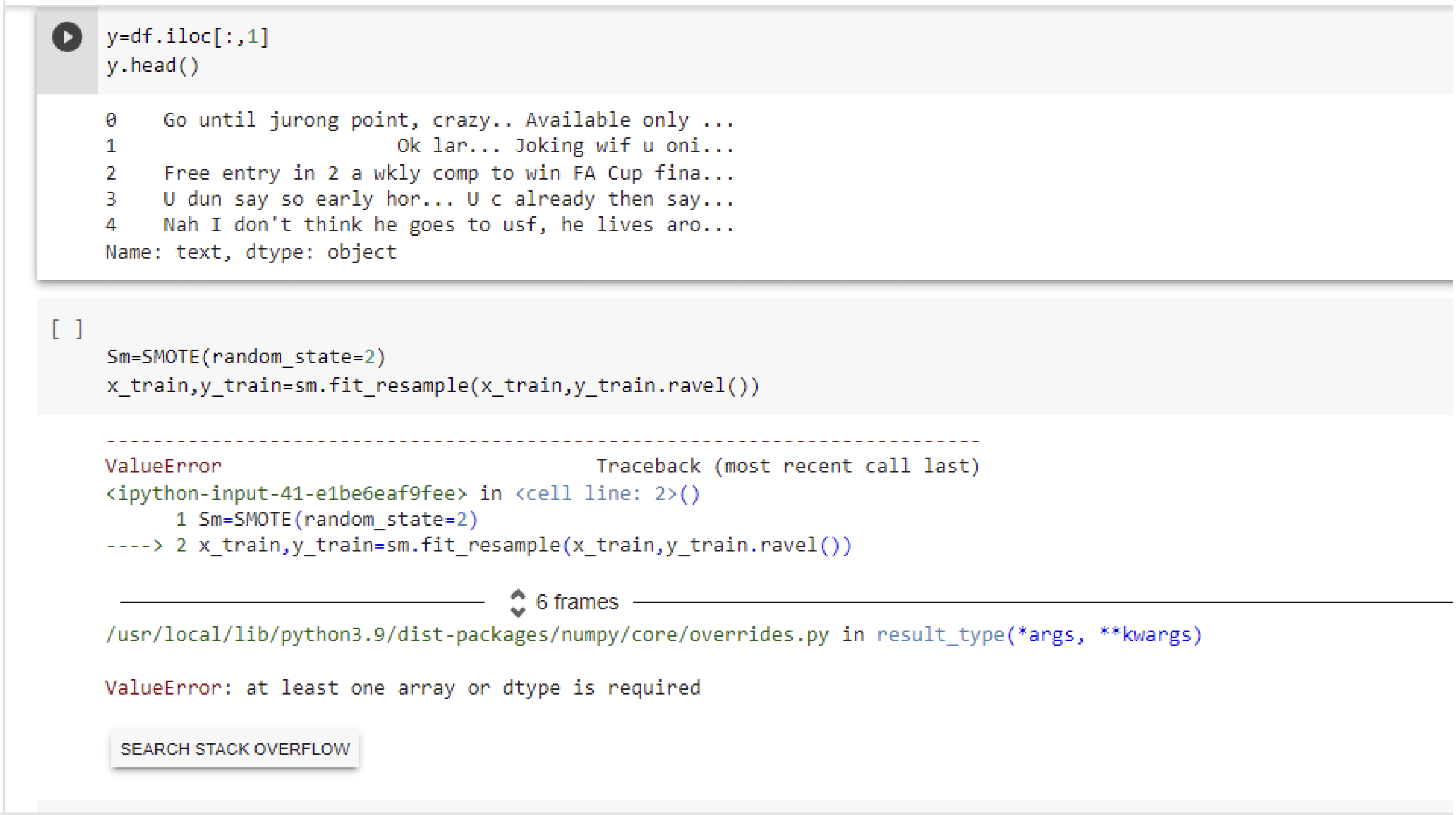
```
After OverSamplings, counts of label'0':0
```

```
from imblearn.over_sampling import SMOTE
```

```
x=df.iloc[:,2:1]
x.head()
```

```
0
1
2
3
4
```

```
y=df.iloc[:,1]
```



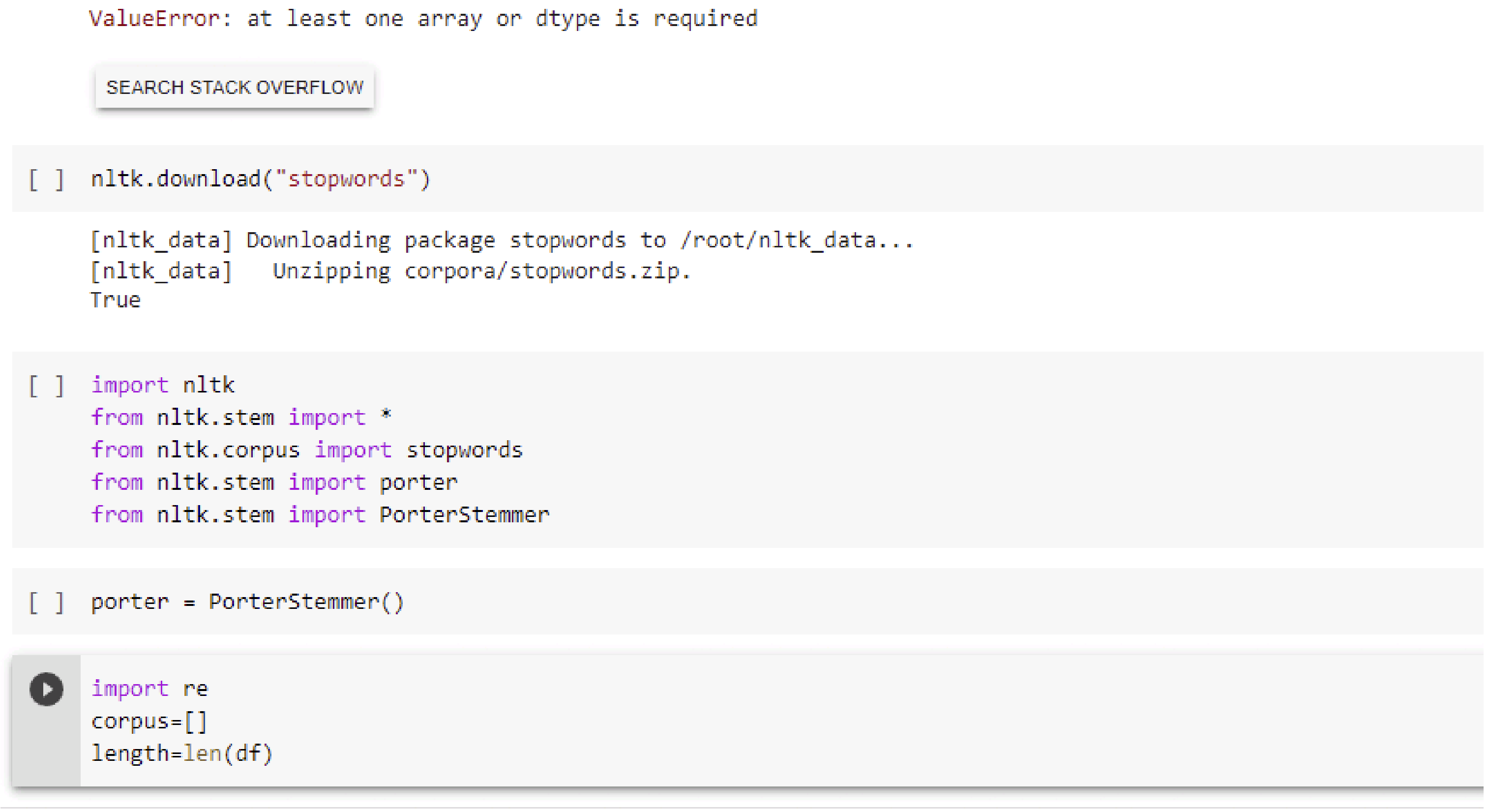
```
y=df.iloc[:,1]
y.head()
```

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

```
Sm=SMOTE(random_state=2)
x_train,y_train=sm.fit_resample(x_train,y_train.ravel())
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-41-e1be6eaf9fee> in <cell line: 2>()
      1 Sm=SMOTE(random_state=2)
----> 2 x_train,y_train=sm.fit_resample(x_train,y_train.ravel())
```

```
/usr/local/lib/python3.9/dist-packages/numpy/core/overrides.py in result_type(*args, **kwargs)

ValueError: at least one array or dtype is required
```



```
ValueError: at least one array or dtype is required
```

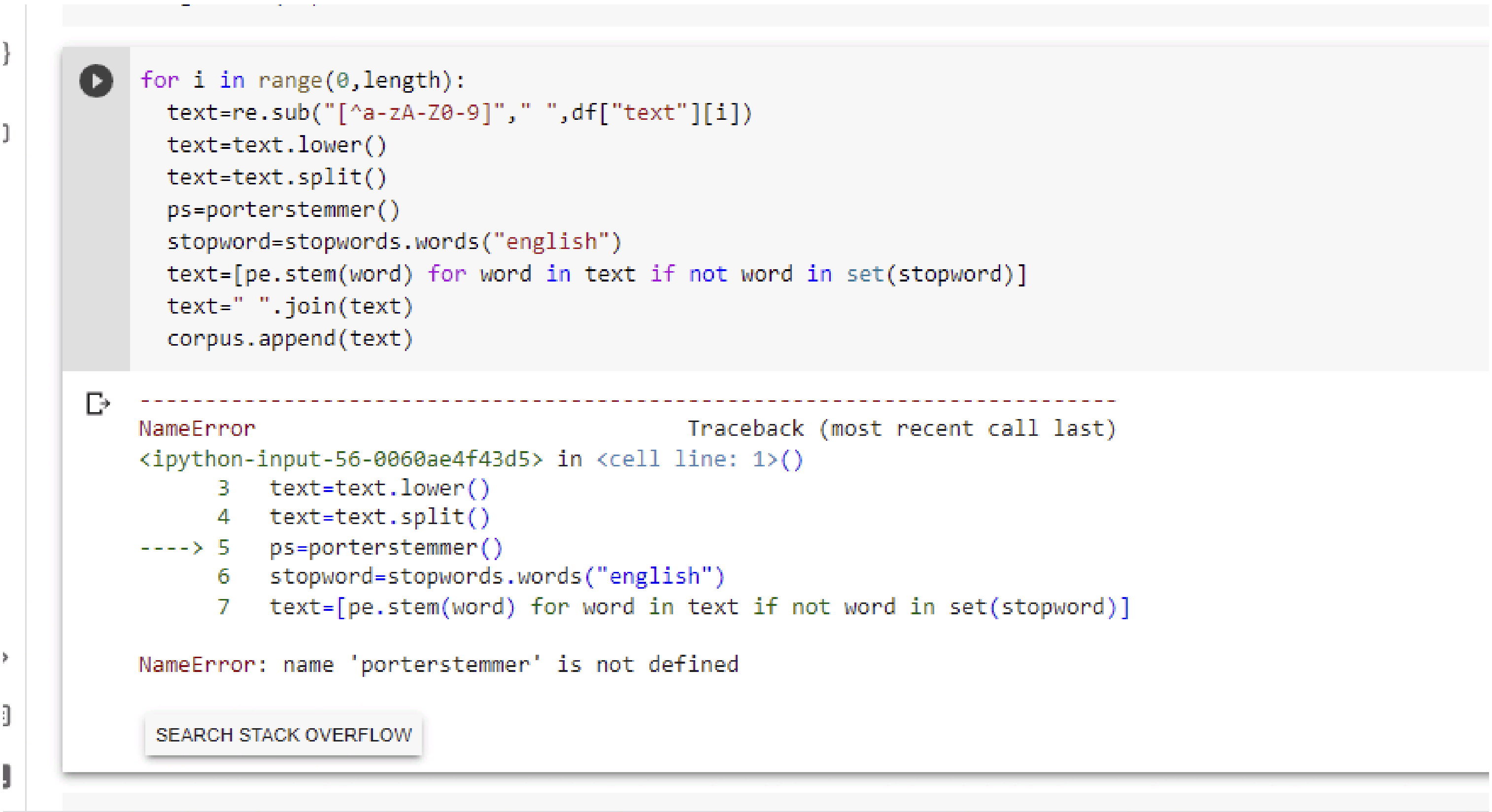
```
[ ] nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
[ ] import nltk
    from nltk.stem import *
    from nltk.corpus import stopwords
    from nltk.stem import porter
    from nltk.stem import PorterStemmer
```

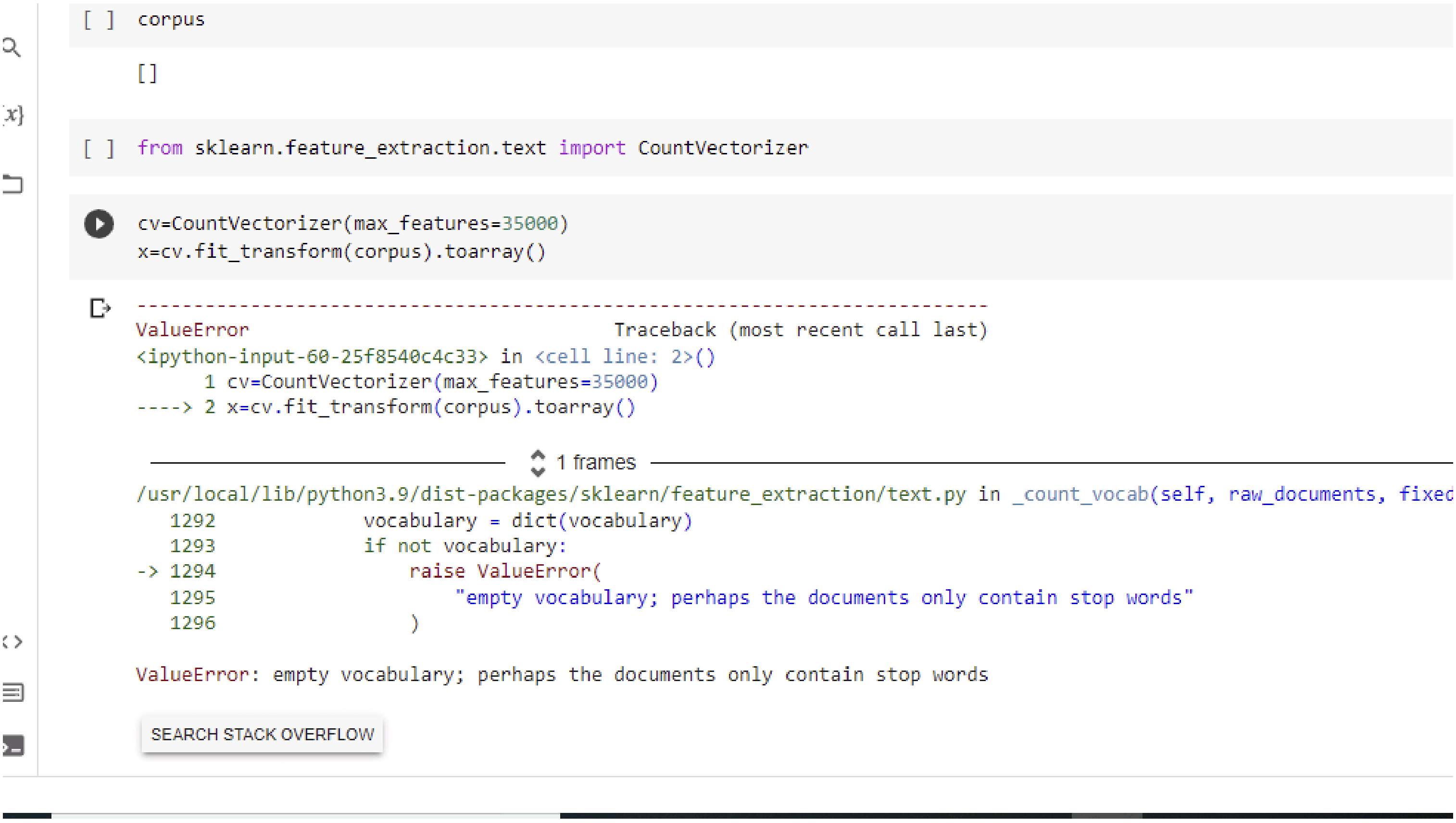
```
[ ] porter = PorterStemmer()
```

```
import re
corpus=[]
length=len(df)
```



```
for i in range(0,length):
  text=re.sub("[^a-zA-Z0-9]"," ",df["text"][i])
  text=text.lower()
  text=text.split()
  ps=porterstemmer()
  stopword=stopwords.words("english")
  text=[pe.stem(word) for word in text if not word in set(stopword)]
  text=" ".join(text)
  corpus.append(text)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-56-0060ae4f43d5> in <cell line: 1>()
      3   text=text.lower()
      4   text=text.split()
----> 5   ps=porterstemmer()
      6   stopword=stopwords.words("english")
      7   text=[pe.stem(word) for word in text if not word in set(stopword)]

NameError: name 'porterstemmer' is not defined
```

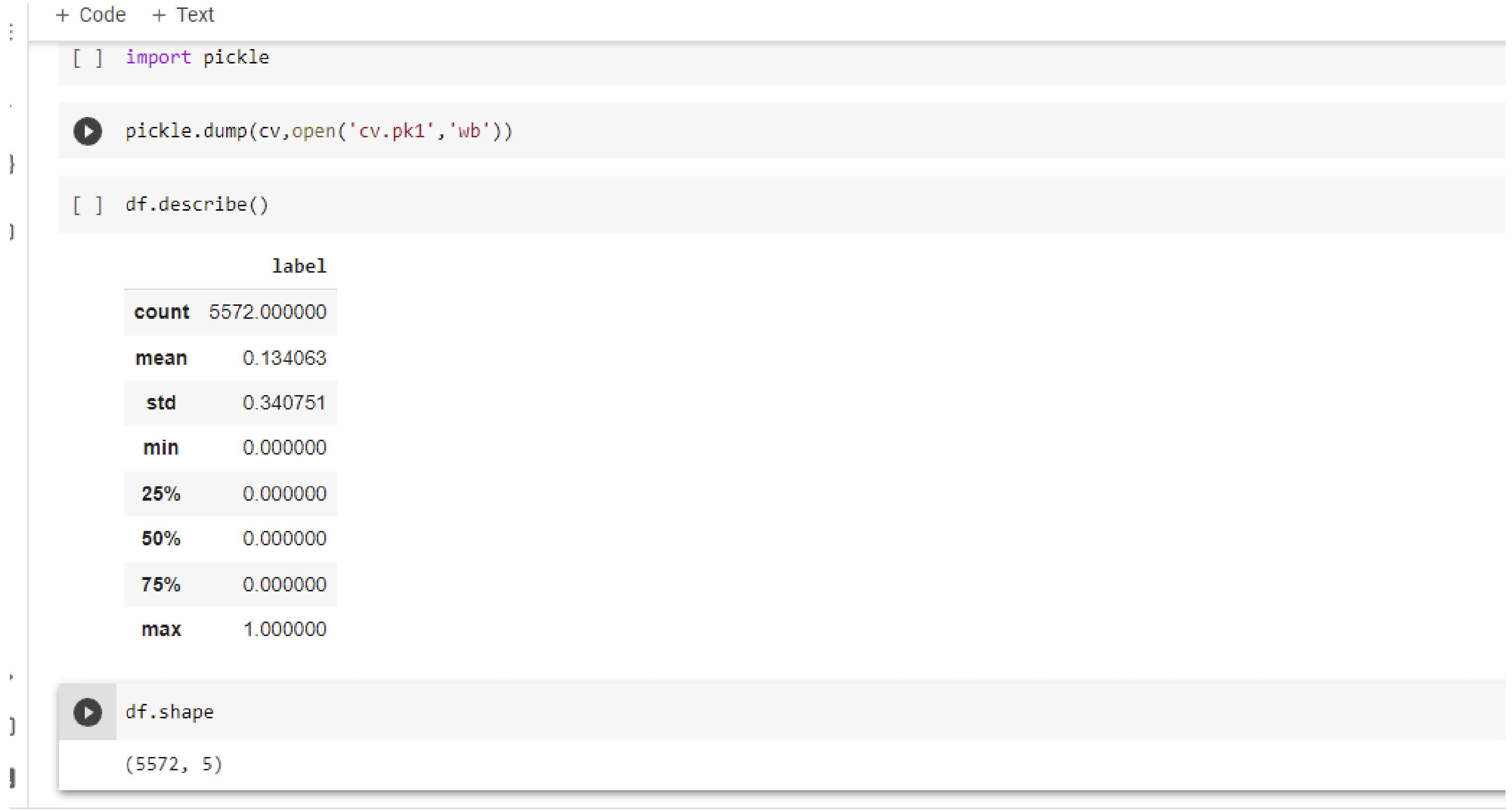


```
[ ] corpus

    []
```

```
[ ] from sklearn.feature_extraction.text import CountVectorizer
```

```
cv=CountVectorizer(max_features=35000)
x=cv.fit_transform(corpus).toarray()
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-60-25f8540c4c33> in <cell line: 2>()
      1 cv=CountVectorizer(max_features=35000)
----> 2 x=cv.fit_transform(corpus).toarray()

                                  1 frames
/usr/local/lib/python3.9/dist-packages/sklearn/feature_extraction/text.py in _count_vocab(self, raw_documents, fixed
   1292             vocabulary = dict(vocabulary)
   1293             if not vocabulary:
-> 1294                 raise ValueError(
   1295                     "empty vocabulary; perhaps the documents only contain stop words"
   1296                 )

ValueError: empty vocabulary; perhaps the documents only contain stop words
```



```
import pickle
```

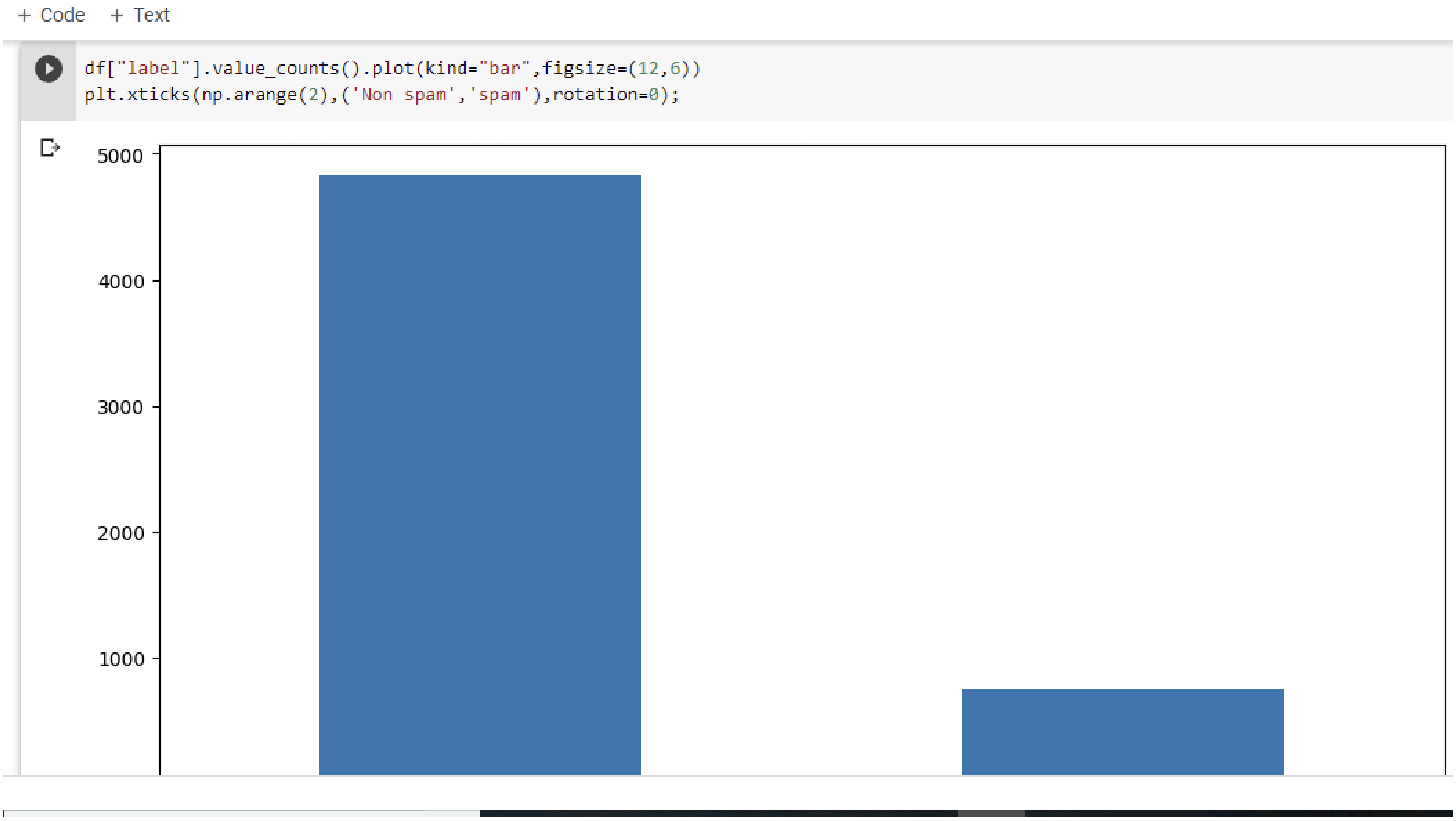
```
pickle.dump(cv,open('cv.pk1','wb'))
```

```
df.describe()
```

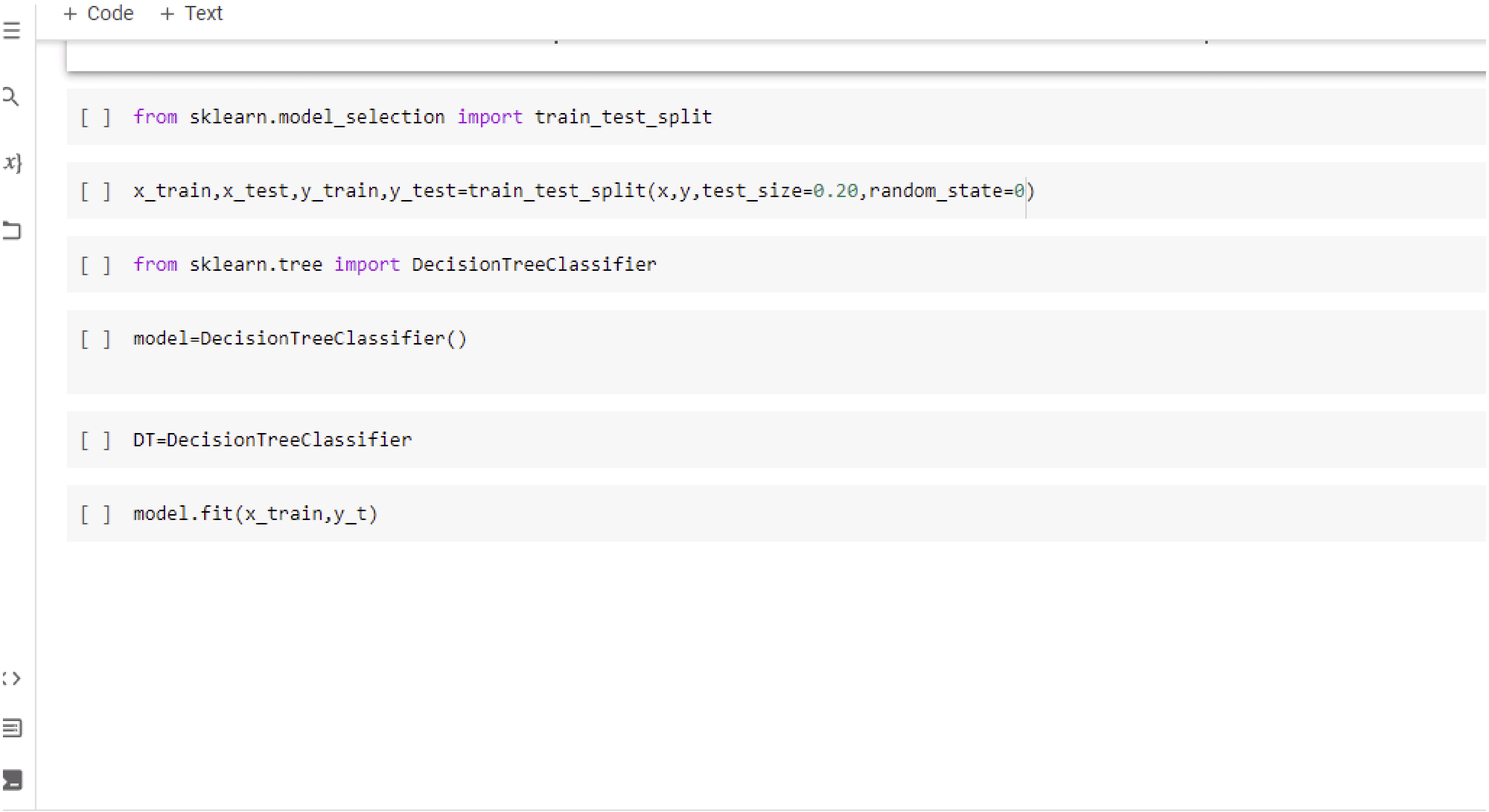
```
                label
count   5572.000000
mean       0.134063
std        0.340751
min        0.000000
25%        0.000000
50%        0.000000
75%        0.000000
max        1.000000
```

```
df.shape
```

```
(5572, 5)
```



```
df["label"].value_counts().plot(kind="bar",figsize=(12,6))
plt.xticks(np.arange(2),('Non spam','spam'),rotation=0);
```



```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)
```

```
from sklearn.tree import DecisionTreeClassifier
```

```
model=DecisionTreeClassifier()
```

```
DT=DecisionTreeClassifier
```

```
model.fit(x_train,y_t)
```