```
#!/usr/bin/env python
# coding:
utf-8

#Dataset-
#https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset

#
In[6]:


import pandas as pd
import numpy as np


# In[7]:


data =
pd.read_csv("SMSSpamCollection.csv",
sep=",",names=["label","message"])


# In[8]:


data


#
In[9]:


data.info()


# In[10]:


data.describe()


# In[11]:


data["label"] =
data["label"].map({'ham':0  ,'spam':1})


# In[12]:


import matplotlib.pyplot as
plt
import seaborn as sns
get_ipython().run_line_magic('matplotlib', 'inline')


#
In[13]:


#Countplot for spam vs
ham
plt.figure(figsize=(4,4))
sns.countplot(x="label",data=data)
plt.title("Coun
tplot")
plt.xlabel("Is the SMS spam?")
plt.ylabel("Count")      #gives
Imbalanced dataset.


# In[14]:
```

```
spam = data[data["label"]==1]
print("No of
spam SMS:", len(spam))
print("No of ham SMS:", len(data)-len(spam))


#
In[15]:


count=int((data.shape[0]-spam.shape[0])/spam.shape[0])
count      #Approx 6 times we
need for doing Balanced data.


# In[16]:


for i in range(0,count-1):

data=pd.concat([data,spam])

data.shape


#
In[17]:


plt.figure(figsize=(4,4))
sns.countplot(x="label",data=data)
plt.title(&quo
t;Countplot")
plt.xlabel("Is the SMS spam?")
plt.ylabel("Count")
#gives Balanced dataset.


# In[18]:


#Extract word from
message
data["word_count"]=data["message"].apply(lambda
x:len(x.split()))
data


# In[19]:


#Plot for ham vs spam w.r.t
distribution
plt.figure(figsize=(8,5))
#(1,1)
plt.subplot(1,2,1)
sns.histplot(data[data["l
abel"]==0].word_count,kde=True)
plt.title("Distribution of Ham
SMS")

#(1,2)
plt.subplot(1,2,2)
sns.histplot(data[data["label"]==1].word_count,
kde=True)
plt.title("Distribution of Spam SMS")

plt.tight_layout()
plt.show()


#
In[20]:
```

```
import nltk
import re
from nltk.corpus import stopwords


# In[22]:


#STOPWORDS =
set(stopwords.words('english'))

def clean_text(text):
    text = text.lower()
    text =
re.sub(r'[^0-9a-zA-Z]', ' ', text)
    text = re.sub(r'\s+', ' ', text)
    #text = "
".join(word for word in text.split() if word not in STOPWORDS)
    return text


#
In[24]:


data['message'] = data['message'].apply(clean_text)


# In[25]:


data


#
In[27]:


#Building Model
x = data['message']
y = data['label']


# In[33]:


from
sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split,
cross_val_score
from sklearn.metrics import classification_report
from
sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer

def
classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.25, random_state=42, shuffle=True, stratify=y)
    pipeline_model =
Pipeline([('vect', CountVectorizer()),

('tfidf',TfidfTransformer()),
                                ('clf', model)])

pipeline_model.fit(x_train, y_train)
    print('Accuracy:', pipeline_model.score(x_test,
y_test)*100)
    y_pred = pipeline_model.predict(x_test)

print(classification_report(y_test, y_pred))


# In[34]:
```

```
#Accuracy using Logistic
Regression
from sklearn.linear_model import LogisticRegression
model =
LogisticRegression()
classify(model, x, y)            #Logistic Regression model has the
accuracy with 98.92


# In[38]:


#Accuracy using Naive-Bayes
from sklearn.naive_bayes import
MultinomialNB
model = MultinomialNB()
classify(model, x, y)       #Naive-bayes model has the
accuracy with 98.62


# In[39]:


#Accuracy using Support Vector Machine
from sklearn.svm
import SVC
model = SVC(C=3)
classify(model, x, y)         #SVC model has the best accuracy with
99.95
```