Problem Statement :- SMS SPAM Classification

Problem Statement:

Over recent years, as the popularity of mobile phone devices has increased, Short

Message Service (SMS) has grown into a multi-billion dollar industry. At the same time,

reduction in the cost of messaging services has resulted in growth in unsolicited

commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS,

Mobile service providers suffer from some sort of financial problems as well as it reduces

calling time for users. Unfortunately, if the user accesses such Spam SMS they may face

the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user

privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of

the major issues in the wireless communication world and it grows day by day.

Perform the Below Tasks to complete the assignment:-

● Download the Dataset:- Dataset

● Import required library

● Read dataset and do pre-processing

● Create Model

● Add Layers (LSTM, Dense-(Hidden Layers), Output)

● Compile the Model

● Fit the Model

● Save The Model

● Test The Model

Perform the Below Tasks to complete the assignment:-

* Required library files

# import libraries for reading data, exploring and plotting  
import numpy as np  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator  
%matplotlib inline# library for train test split  
from sklearn.model\_selection import train\_test\_split# deep learning libraries for text pre-processing  
import tensorflow as tf  
from tensorflow.keras.preprocessing.text import Tokenizer  
from tensorflow.keras.preprocessing.sequence import pad\_sequences# Modeling   
from tensorflow.keras.callbacks import EarlyStopping  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Embedding, GlobalAveragePooling1D, Dense, Dropout, LSTM, Bidirectional

● Read dataset and do pre-processing

Data preprocessing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So for this, we use data preprocessing task.

● Create Model

The models are as below:

1. Multinomial Naive Bayes Model (Count tokenizer)
2. Multinomial Naive Bayes Model (tfidf tokenizer)
3. Support Vector Classifier Model
4. Logistic Regression Model with ngrams parameters

● Add Layers (LSTM, Dense-(Hidden Layers), Output)

Human uses short message service (SMS) in mobile as a way of communication or business. Recently SMS is the most used data service in the world. The world sent 8.3 trillion SMS messages in the year 2017, the number of SMS messages sent monthly is 690 billion, so SMS is important for business communications . Recently, SMS spam target mobile phones. SMS spam refers to any illusion text message that is delivered using the mobile network. They are disturbing to users . A survey exposes that 68% of mobile phone users are affected by SMS Spam . In some cases, SMS spam contains malicious activities, such as smishing. Smishing is a cyber-security attack for mobile user aimed at deceiving the user via SMS spam messages that may include a link or malicious software or both. Smishing is combined of two words: SMS and Phishing

● Compile the Model

Compile the model

model provides a method, **compile()** to compile the model. The argument and default value of the **compile()** method is as follows

compile(

optimizer,

loss = None,

metrics = None,

loss\_weights = None,

sample\_weight\_mode = None,

weighted\_metrics = None,

target\_tensors = None

)

The important arguments are as follows −

* loss function
* Optimizer
* metrics

A sample code to compile the mode is as follows −

import losses

import optimizers

import metrics

model.compile(loss = 'mean\_squared\_error',

optimizer = 'sgd', metrics = [metrics.categorical\_accuracy])

where,

* loss function is set as **mean\_squared\_error**
* optimizer is set as **sgd**
* metrics is set as **metrics.categorical\_accuracy**

● Fit the Model

Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. A model that is well-fitted produces more accurate outcomes. A model that is overfitted matches the data too closely. A model that is underfitted doesn’t match closely enough.

Each machine learning algorithm has a basic set of parameters that can be changed to improve its accuracy During the fitting process, you run an algorithm on data for which you know the variable, known as “labeled” data, and produce a machine learning model. Then, you compare the outcomes to real, observed values of the target variable to determine their accuracy.

● Save The Model

Model progress can be saved during and after training. This means a model can resume where it left off and avoid long training times. Saving also means you can share your model and others can recreate your work. When publishing research models and techniques, most machine learning practitioners share:

* code to create the model, and
* the trained weights, or parameters, for the model

Sharing this data helps others understand how the model works and try it themselves with new data.

● Test The Model

* Train your model for a few iterations and verify that the loss decreases.
* Train your algorithm without regularization. If your model is complex enough, it will memorize the training data and your training loss will be close to 0.
* Test specific subcomputations of your algorithm.