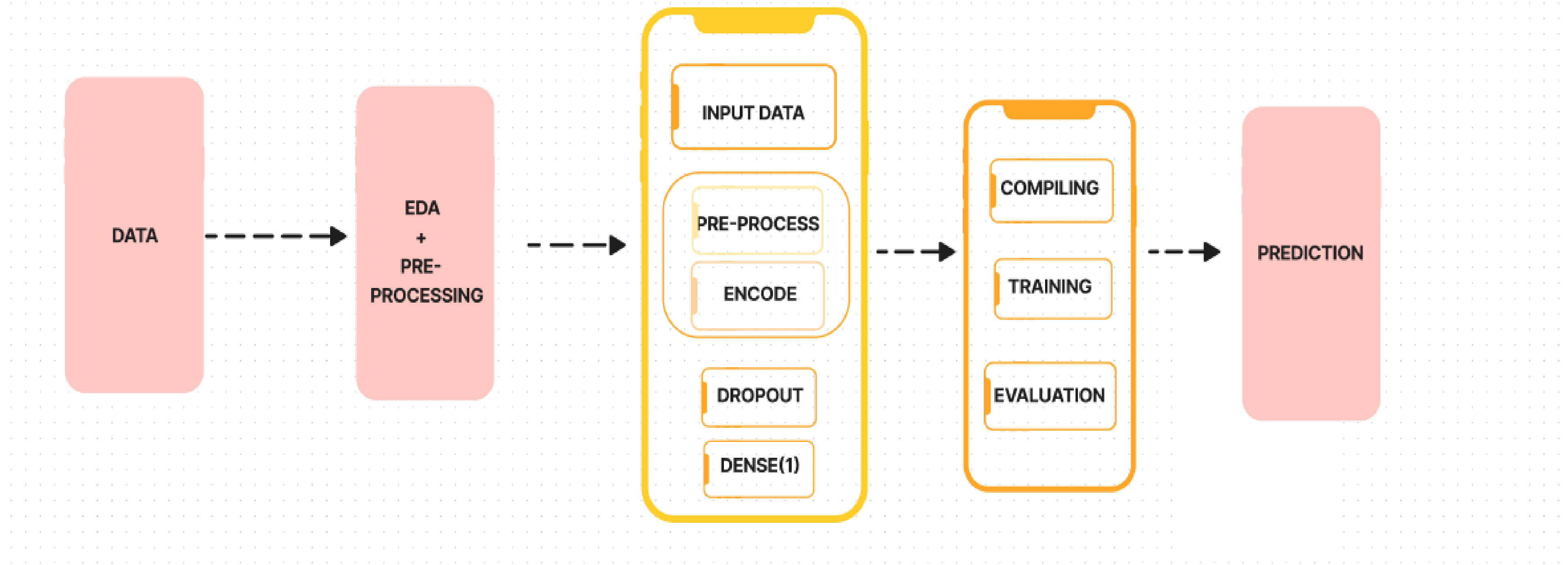
Project : Building a Smarter AI-Powered Spam Classifier

Phase 2 : Explore innovative techniques and approaches to building our spam classifier

Introduction :

BERT, which stands for "*Bidirectional Encoder Representations from Transformers*," is a state-of-the-art natural language processing (NLP) model developed by Google. BERT is designed for feature extraction. These models have demonstrated superior performance in NLP tasks.

PROJECT DESIGN



Steps for Spam Detection Using BERT:

To build the system ourselves we are going to follow these procedures:

1. Load Data – We will be loading our data which is simple [2 categories(ham and spam) along with corresponding emails] CSV file. The file can be found here

2. EDA – Perform some EDA to get a feel of what data looks like – statistics here!

3. Pre Processing – Based on the results of EDA we will be going to apply some preprocessing to the data to make it model friendly

4 – Model Creation – We will use Functional API* to build our Deep Learning Model which will consist of

- INPUT – Will create an input layer that will take in all the pre-processing data and pass it to bert_processor.
- BERT PREPROCESSOR – Process our input text to be in the format which encoder accepts. (adds MASK AND ENCODINGS).
- BERT ENCODER – Feed our processed text to the bert model which will eventually generate a contextualized word embedding for our training data.

- **DROPOUT – Add a dropout layer that will randomly drop out a few neurons from the network to take care overfitting problem.**
- **SOFTMAX – At last we will add a softmax layer to predict data in 2 categories.**
- **+ 2 more generic steps.**

5- Model Evaluation – Further we will check how the model performs on the test data and plot some relevant charts and metrics based on the observations.

6- Predict Data – Finally we will predict our own emails using the model.

Implementing Spam Detection Using BERT:

- **To start we will first readily downloaded dataset which can be from the below link,**

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

- **Loading Dependencies:**

  So, first of all, we are going to import few libraries namely:

  - **Tensorflow_hub**: Place where all TensorFlow pre-trained models are stored.
  - **Tensorflow:** For model creation

- **Pandas:** For data loading, manipulation and wrangling.
- **Tensorflow_text**: Allows additional NLP text processing capabilities outside the scope of tensorflow.
- **Skelarn:** For doing data splitting
- **Matplotlib**: For visualization support

Loading Data

Now we will just load our data into a pandas dataframe ‘df‘ **using its** read_csv() **method/fn and view the first 5 samples using the** head() **method:**

```
# load data
df = pd.read_csv('/content/spam_data.csv')
df.head()
```

Spam Detection Exploratory Data Analysis

We will now do some of the Exploratory – Data Analysis to check how data is distributed along 2 categories. This will give us a feel if we need to do some type of preprocessing over data or is it on the same scale.

To perform this operation we will just be **grouping the data based on category** and call **value_counts()** method on it like:

```
# check count and unique and top values and their frequency
```

```
df['Category'].value_counts()
```

Downsampling Data

Downsampling is a technique where the majority class is downsampled to match the minority class. Since our data has only one column(feature) it ok to use it.

1) We first calculated the percentage of data that needs to be balanced by **dividing minority (spam) by majority(ham)** :

We perform downsampling by just picking any random 747 samples from the ham class. Here is the code to do that:

1) We first calculated the percentage of data that needs to be balanced by **dividing minority (spam) by majority(ham)** :

```
# check percentage of data - states how much data needs to be balanced
print(str(round(747/4825,2))+'%')
>> 0.15%
```

It came out to be 15% of data that needs to be balanced.

2) We then create 2 new datasets namely ham and spam and filtered the data having categories as spam and ham and append it to the respective dataset and finally printed their shape to confirm the filtering and creation

3) Now we will sample the ham dataset using the sample() method with the shape of our spam dataset and to be more specific load it into a new dataframe **df_ham_downsampled and** print its shape to cross verify.

4)Finally we will **concatenate** our **df_ham_downsampled** and **df_spam** to create a final dataframe called **df_balalnced.**

Preprocessing of Spam Detection Data

<u>One Hot Encoding Categories</u>

As can be seen, we have only text as categorical data, and the model doesn't understand them. So instead of text, we can just assign integer labels to our class **ham and spam as 0 and 1 respectively,** and store it in new column **spam. This is called- Hot-Encoding**

To achieve this we will just be filtering the column category and perform operations:

- 1 – If a category is a ham/ not spam
- 0 – if the category is spam

Model Creation

- _To create our model we will first **download the bert preprocessor and encoder**(for more info refer to the previous article) as it allows us to use them as **function pointers** where one can feed our inputs and get the **processed output and embedding**. Also, this helps in better **readability** of the code**.**

Model Evaluation

To evaluate our model we will simply use the **model's evaluate** method feeding it **testing data and labels** to get a rough estimate of how the model is performing.

Endnote

1. A **systematic procedure** to work on Deep Learning Projects.
2. Some of the Data Science concepts such as *file i/o using pandas, down-sampling, confusion matrix, classification reports, and plotting them using seaborn*.
3. How to use **functional API to** create complex models.
4. How to **load and use the BERT model** very easily(literally in 5 lines of code).
5. And lastly got some info in the world of spams and spammers.