# Predicting Geolocation from Tweets

Yachay.ai Externship

# Overview

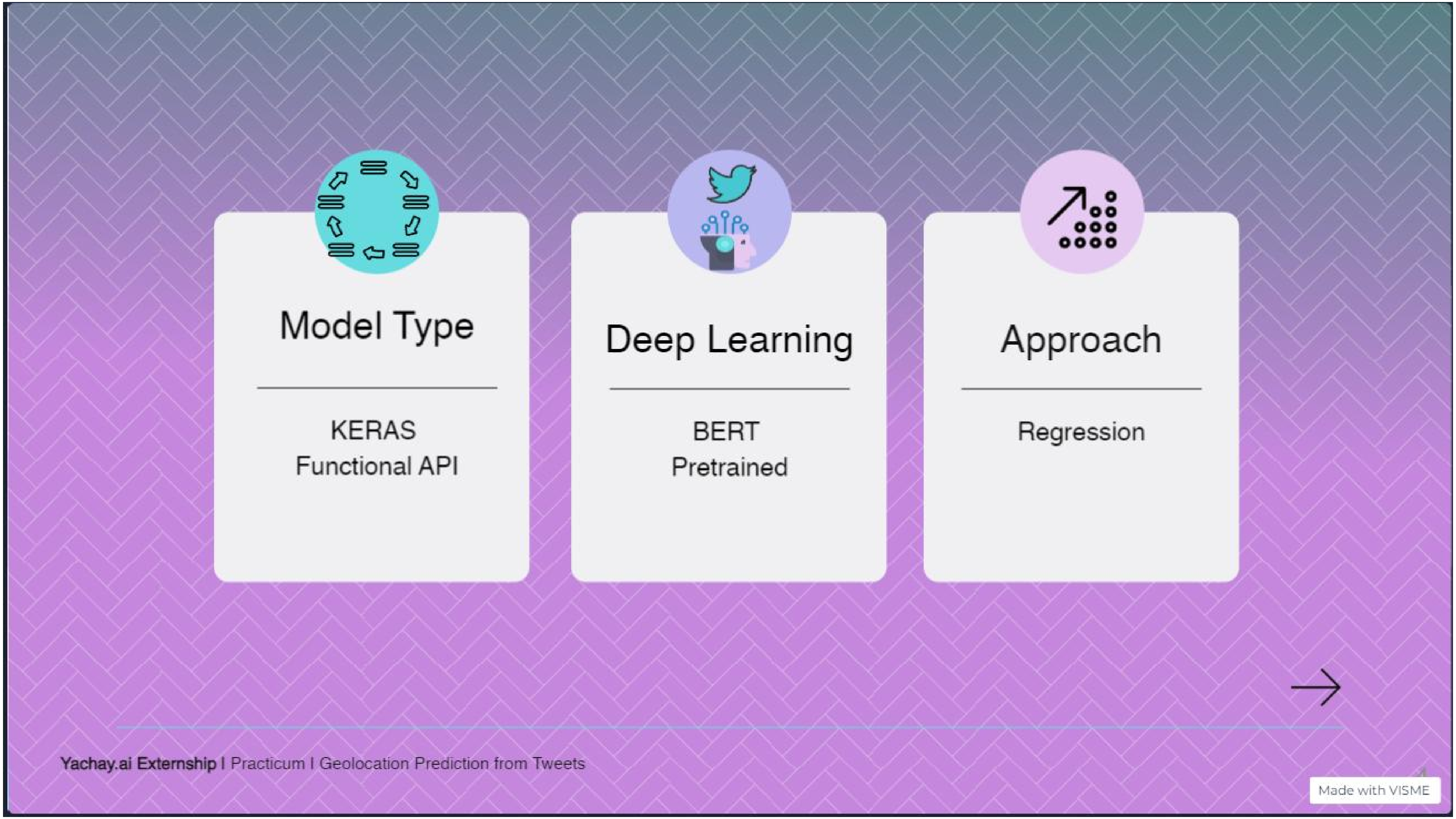
→

01 Model Selection

02 Preprocessing

03 Results

# Model Selection



## Model Type

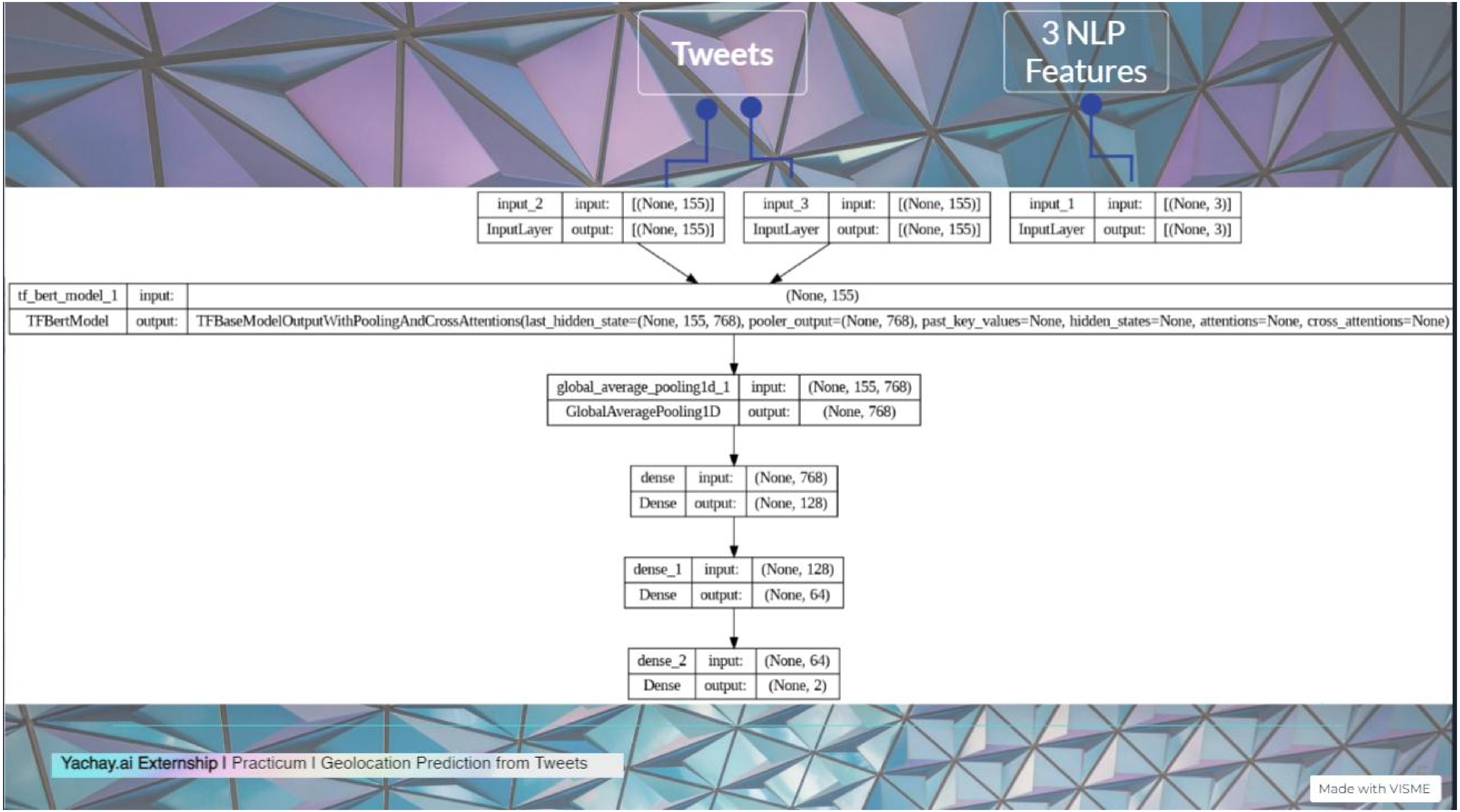
KERAS
Functional API

## Deep Learning

BERT
Pretrained

## Approach

Regression



Tweets

3 NLP Features

| input_2 | input: | [(None, 155)] |
|---|---|---|
| InputLayer | output: | [(None, 155)] |

| input_3 | input: | [(None, 155)] |
|---|---|---|
| InputLayer | output: | [(None, 155)] |

| input_1 | input: | [(None, 3)] |
|---|---|---|
| InputLayer | output: | [(None, 3)] |

| tf_bert_model_1 | input: | (None, 155) |
|---|---|---|
| TFBertModel | output: | TFBaseModelOutputWithPoolingAndCrossAttentions(last_hidden_state=(None, 155, 768), pooler_output=(None, 768), past_key_values=None, hidden_states=None, attentions=None, cross_attentions=None) |

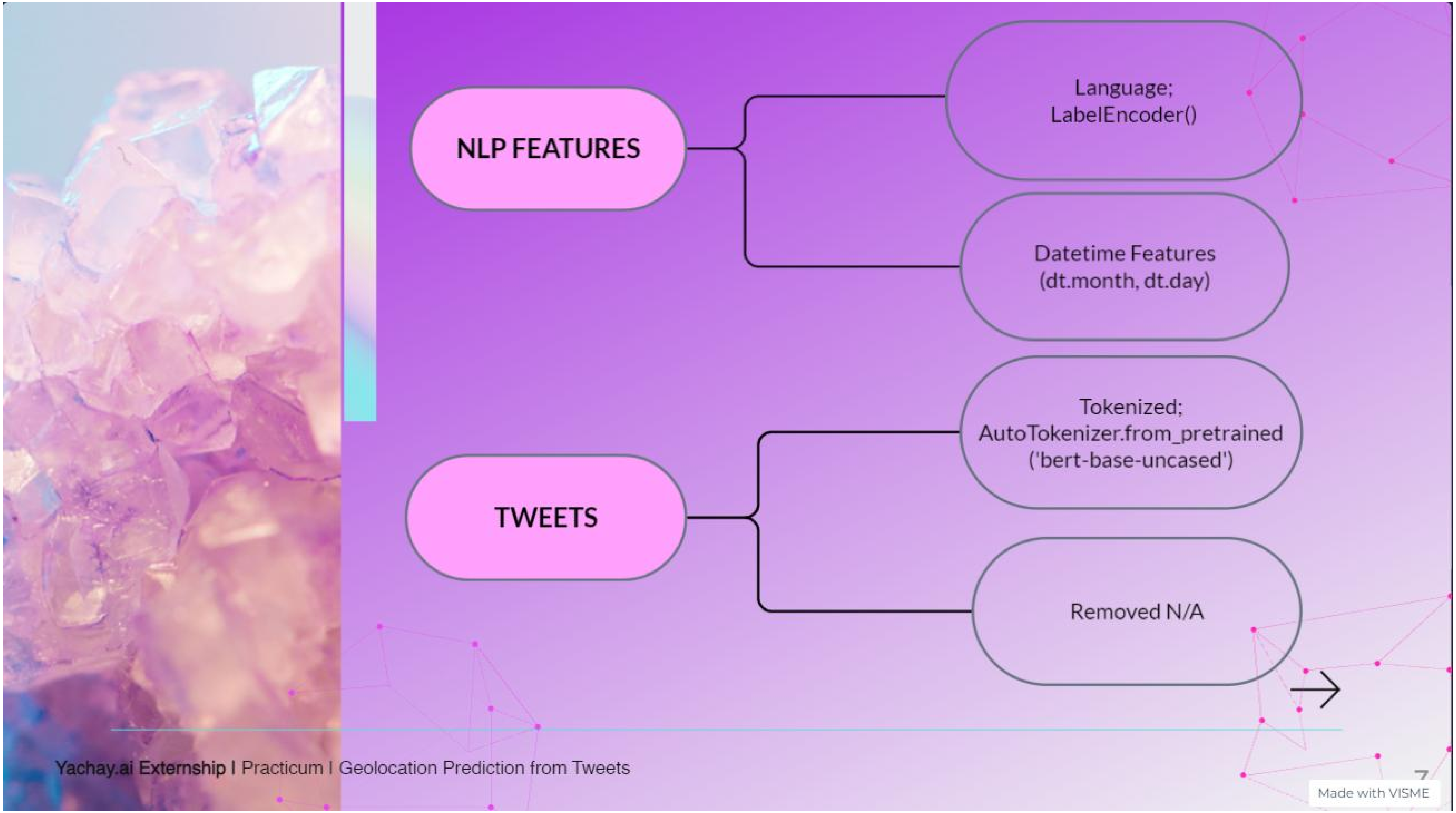
| global_average_pooling1d_1 | input: | (None, 155, 768) |
|---|---|---|
| GlobalAveragePooling1D | output: | (None, 768) |

| dense | input: | (None, 768) |
|---|---|---|
| Dense | output: | (None, 128) |

| dense_1 | input: | (None, 128) |
|---|---|---|
| Dense | output: | (None, 64) |

| dense_2 | input: | (None, 64) |
|---|---|---|
| Dense | output: | (None, 2) |

# Preprocessing

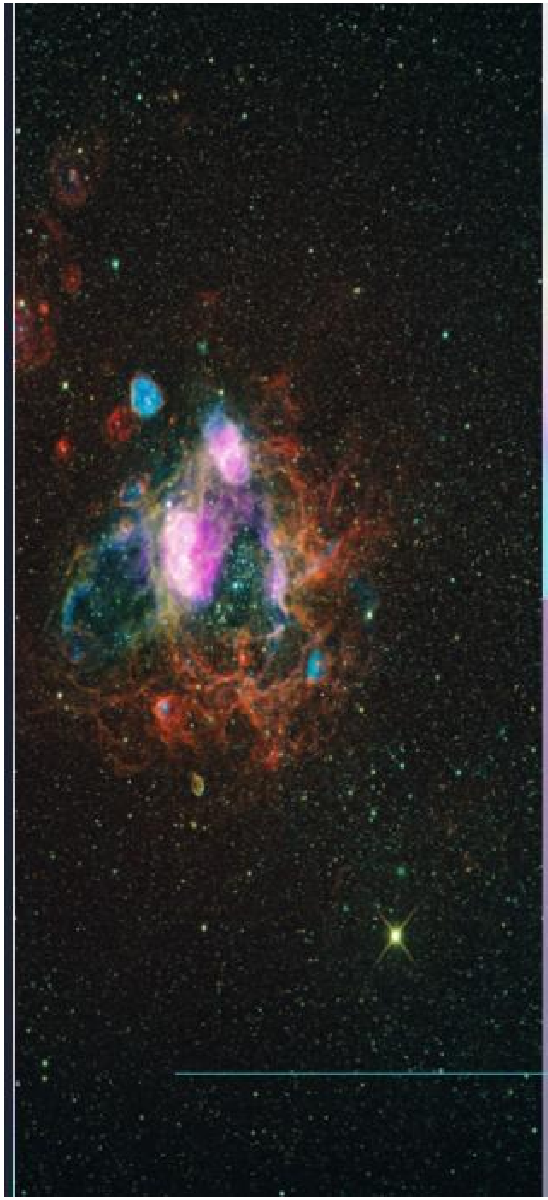


NLP FEATURES

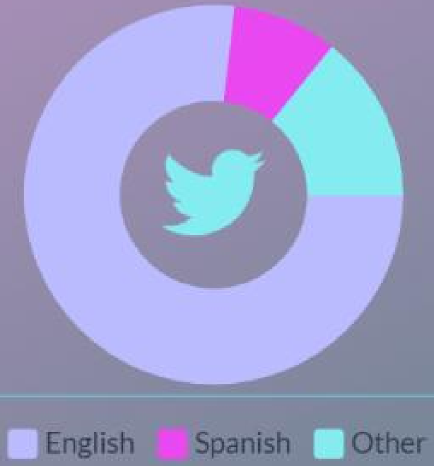
- Language; LabelEncoder()
- Datetime Features (dt.month, dt.day)

TWEETS

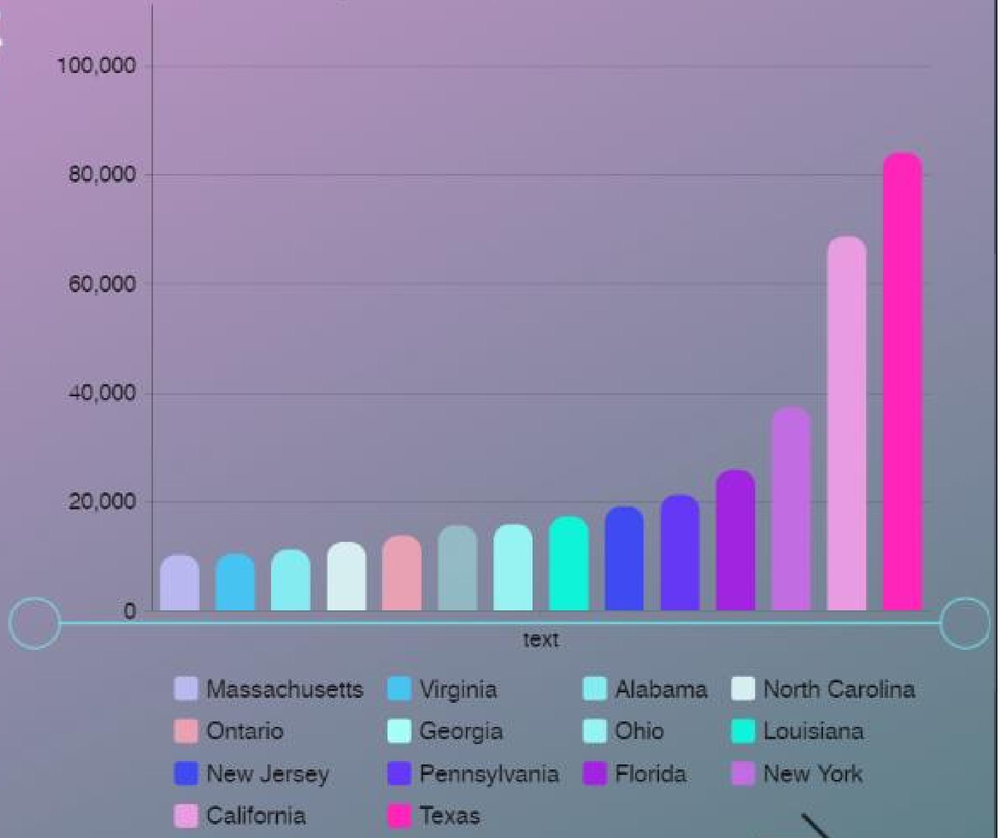
- Tokenized; AutoTokenizer.from_pretrained ('bert-base-uncased')
- Removed N/A



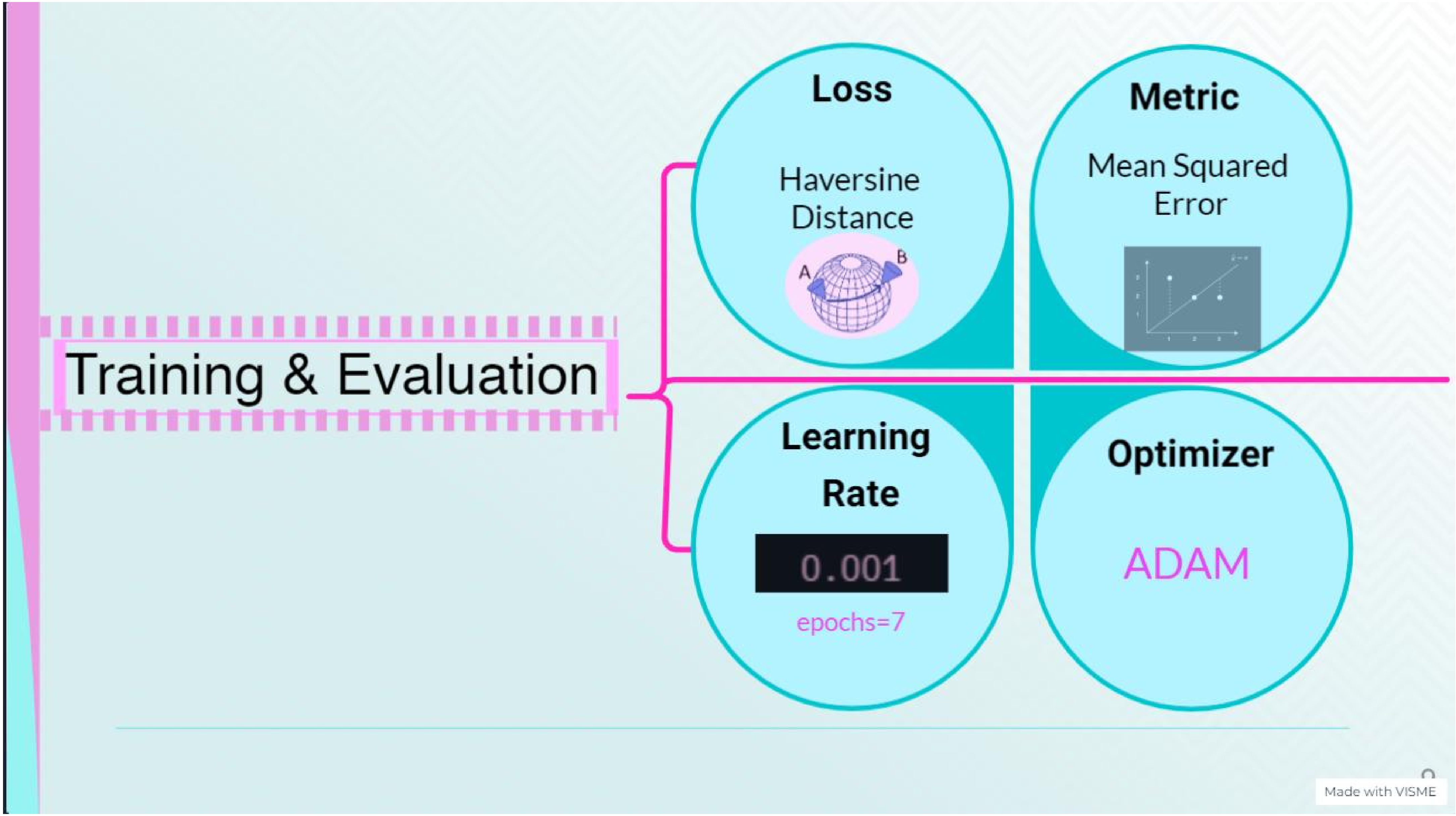
## Language Distribution



## Regions > 10,000 Tweets



**Yachay.ai Externship** | Practicum | Geolocation Prediction from Tweets



# Training & Evaluation

**Loss**

Haversine Distance

**Metric**

Mean Squared Error

**Learning Rate**

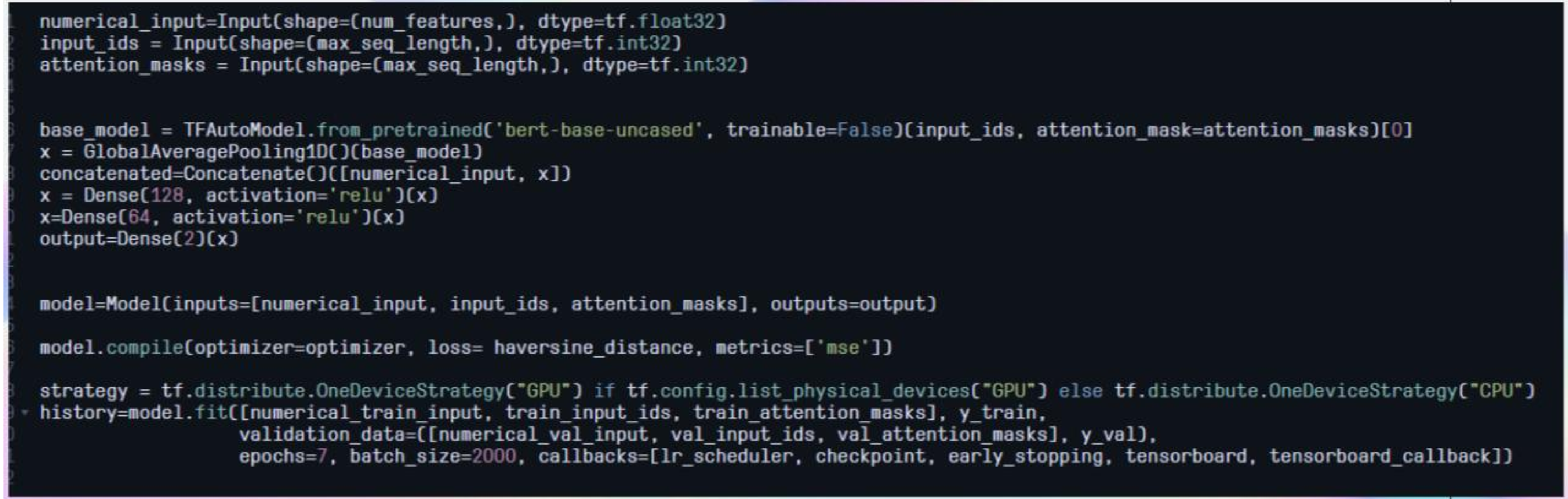
0.001

epochs=7

**Optimizer**

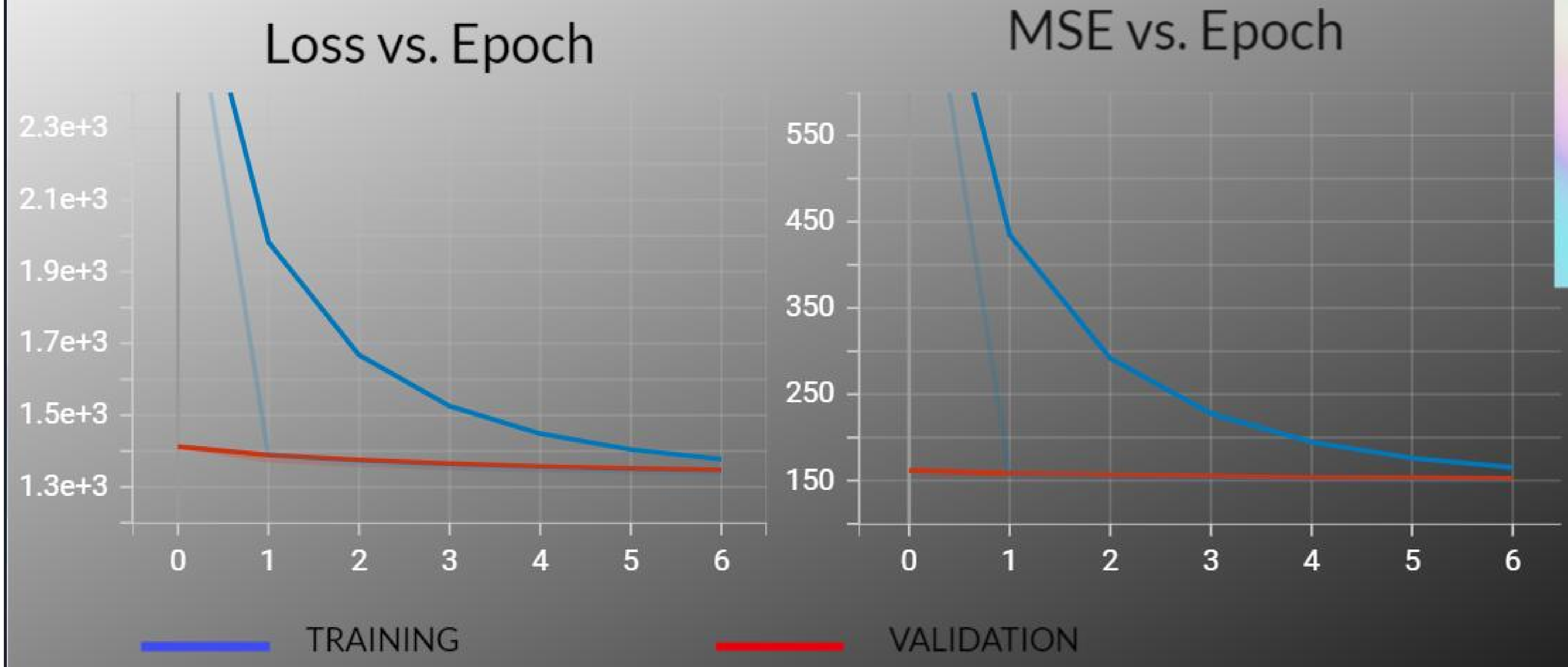
ADAM

# Model Structure



```
numerical_input=Input(shape=(num_features,), dtype=tf.float32)
input_ids = Input(shape=(max_seq_length,), dtype=tf.int32)
attention_masks = Input(shape=(max_seq_length,), dtype=tf.int32)


base_model = TFAutoModel.from_pretrained('bert-base-uncased', trainable=False)(input_ids, attention_mask=attention_masks)[0]
x = GlobalAveragePooling1D()(base_model)
concatenated=Concatenate()([numerical_input, x])
x = Dense(128, activation='relu')(x)
x=Dense(64, activation='relu')(x)
output=Dense(2)(x)


model=Model(inputs=[numerical_input, input_ids, attention_masks], outputs=output)

model.compile(optimizer=optimizer, loss= haversine_distance, metrics=['mse'])

strategy = tf.distribute.OneDeviceStrategy("GPU") if tf.config.list_physical_devices("GPU") else tf.distribute.OneDeviceStrategy("CPU")
history=model.fit([numerical_train_input, train_input_ids, train_attention_masks], y_train,
                  validation_data=([numerical_val_input, val_input_ids, val_attention_masks], y_val),
                  epochs=7, batch_size=2000, callbacks=[lr_scheduler, checkpoint, early_stopping, tensorboard, tensorboard_callback])
```

# Results



Loss vs. Epoch

MSE vs. Epoch

# Results

| DATA | LOSS | MSE |
|---|---|---|
| TRAINING | 1356 km | 153 km$^2$ |
| VALIDATION | 1345 km | 152 km$^2$ |
| TEST | 1334 km | 149 km$^2$ |

# Final Reflection

## Limitations

□ Validation data outperformed

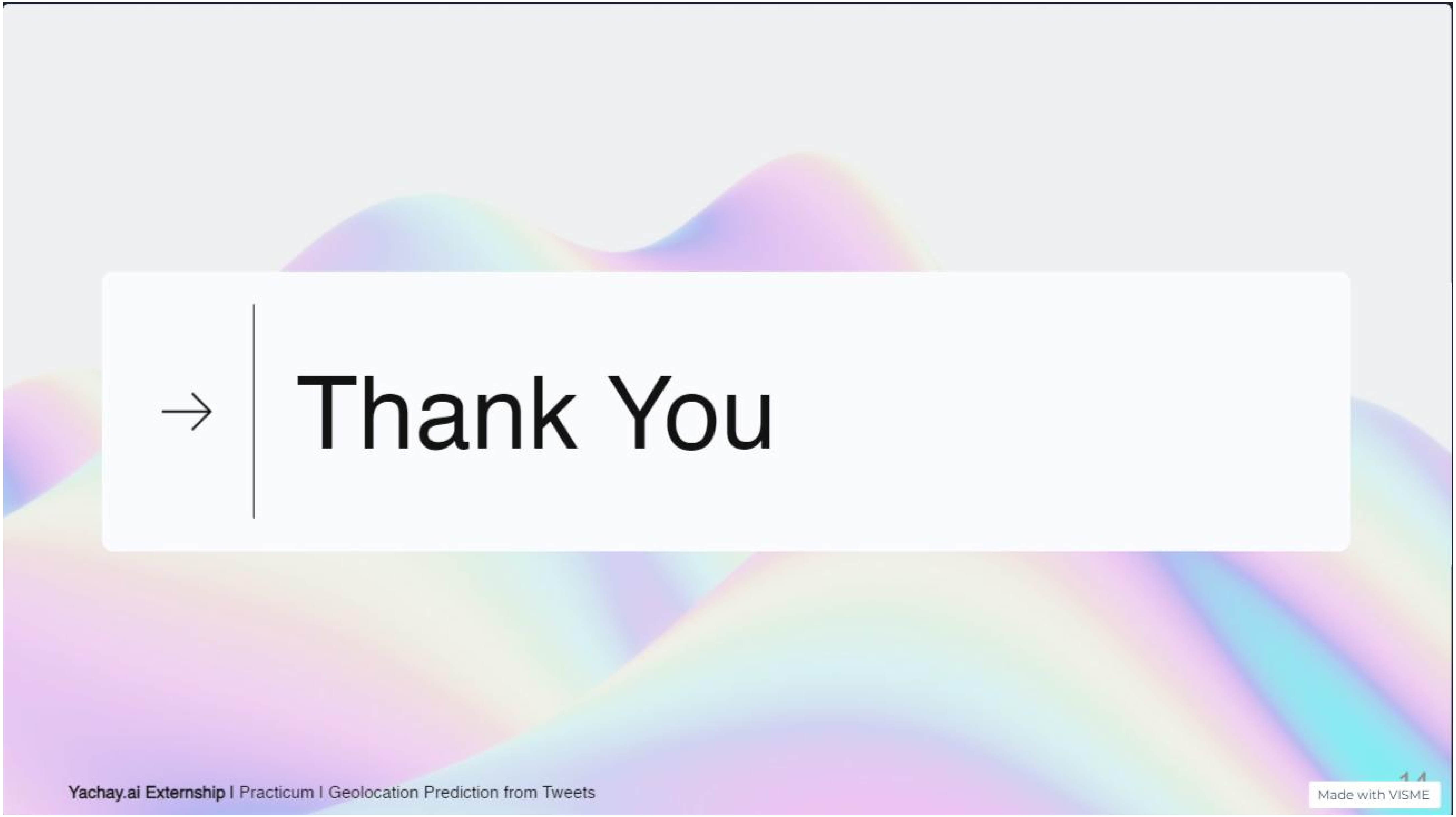
□ Short Training

## Strengths

□ Utilizes BERT

□ NLP Features

□ Convenient & Buildable



# → Thank You