A Report on

Text Classification with TensorFlow

Submitted by

Mr. DHARSHAN S

Under the Guidance of

Mr. Kishore Kumar

E-Mail ID:
dharshans805@gmail.com

Mobile No:
9739512397

Text Classification with TensorFlow

❖ Abstract:

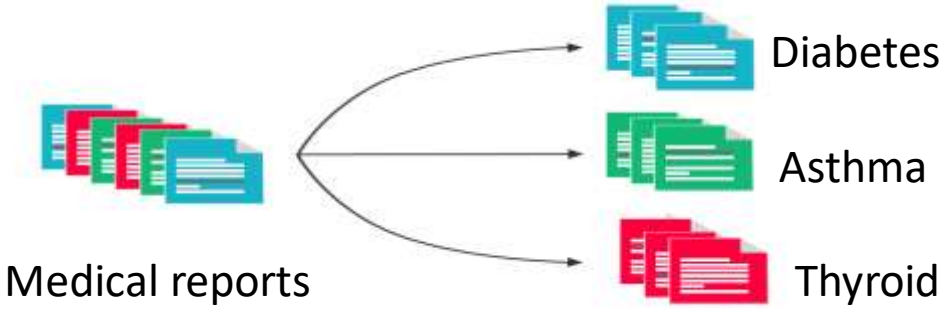
This report presents a project on text classification using TensorFlow, focusing on predicting medical conditions based on Medical data. The project involves preprocessing text data, encoding labels, tokenizing the text, and building a neural network model for classification. The model is trained and evaluated to determine its accuracy.

❖ Objective:

The objective of this project is to build and evaluate a text classification model using TensorFlow to predict medical conditions based on medical information. The project aims to preprocess the data, create a suitable neural network architecture, and achieve a high accuracy in predictions.

❖ Introduction:

Text classification is a common task in natural language processing (NLP) that involves categorizing text into predefined classes. In the medical field, text classification can be used to predict conditions based on various text inputs such as symptoms, patient history, or demographic information. This project utilizes gender data to predict medical conditions using a deep learning model implemented with TensorFlow.



TensorFlow, an open-source machine learning library developed by Google, provides powerful tools for building and training text classification models.

➢ Key Concepts in Text Classification:

1. Preprocessing:

1. **Tokenization**: Splitting text into individual words or tokens.
2. **Padding**: Ensuring all sequences have the same length by adding zeros to shorter sequences.
3. **Encoding**: Converting categorical labels into numerical values.

2.Model Building:

1. **Embedding Layer**: Converts words into dense vectors of fixed size, capturing semantic information.
2. **Dense Layers**: Fully connected layers that learn features from the input data.

3. Training:

1. **Optimizer**: Algorithm to adjust model weights (e.g., Adam).
2. **Loss Function**: Measures the difference between predicted and actual labels (e.g., binary cross-entropy).

4. Evaluation:

Assessing model performance using metrics like accuracy on a separate test set.

❖ Methodology:

1.Data Loading: The medical data is loaded from an Excel file.

2.Data Preprocessing: The gender text data and corresponding condition labels are extracted.

3.Label Encoding: The condition labels are encoded into numerical format using Label Encoder.

4.Text Tokenization: The gender text data is tokenized, converting text into sequences of integers.

5.Sequence Padding: The tokenized sequences are padded to ensure uniform length.

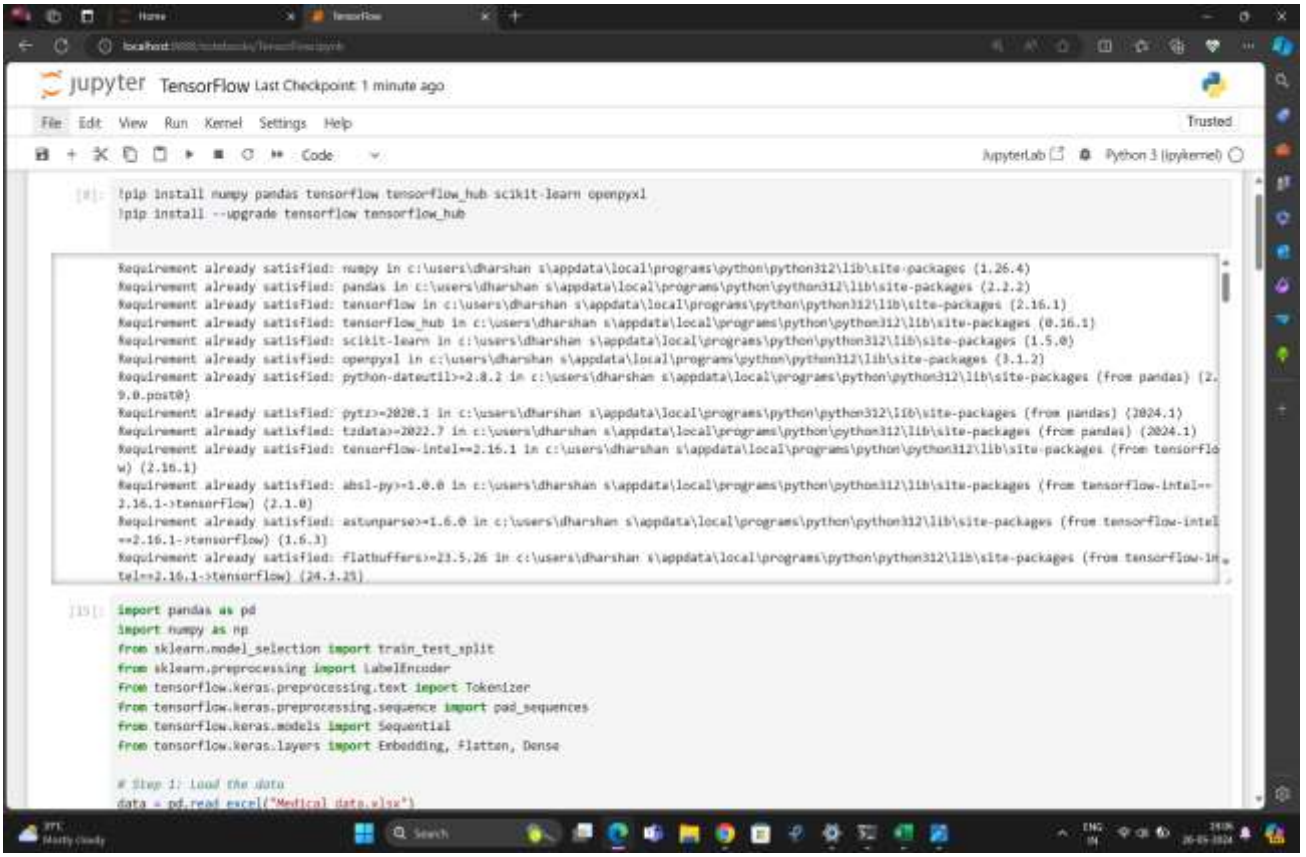
6.Model Building: A Sequential neural network model is built with an Embedding layer, Flatten layer, and Dense layers.

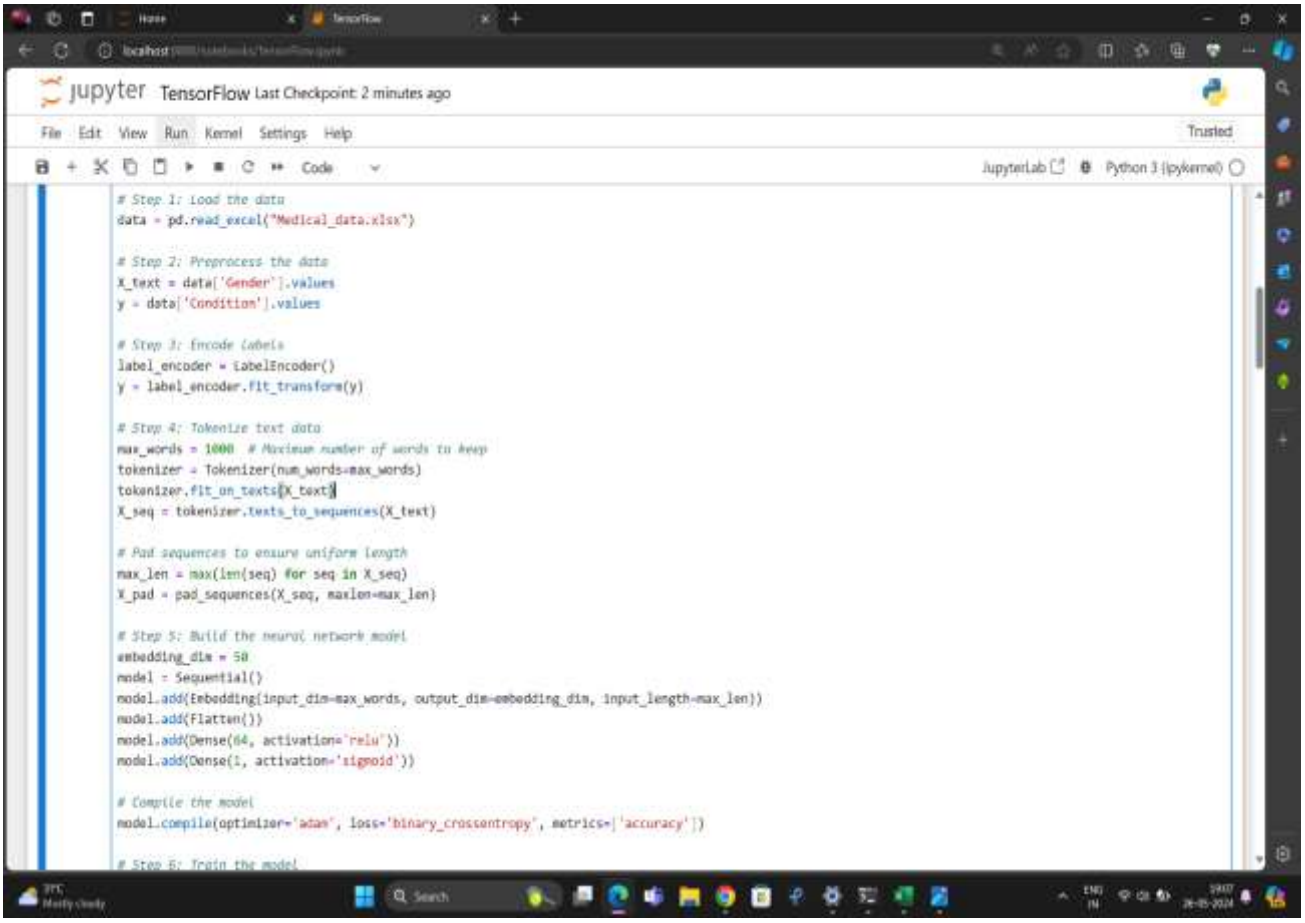
7.Model Compilation: The model is compiled with the Adam optimizer and binary cross-entropy loss function.

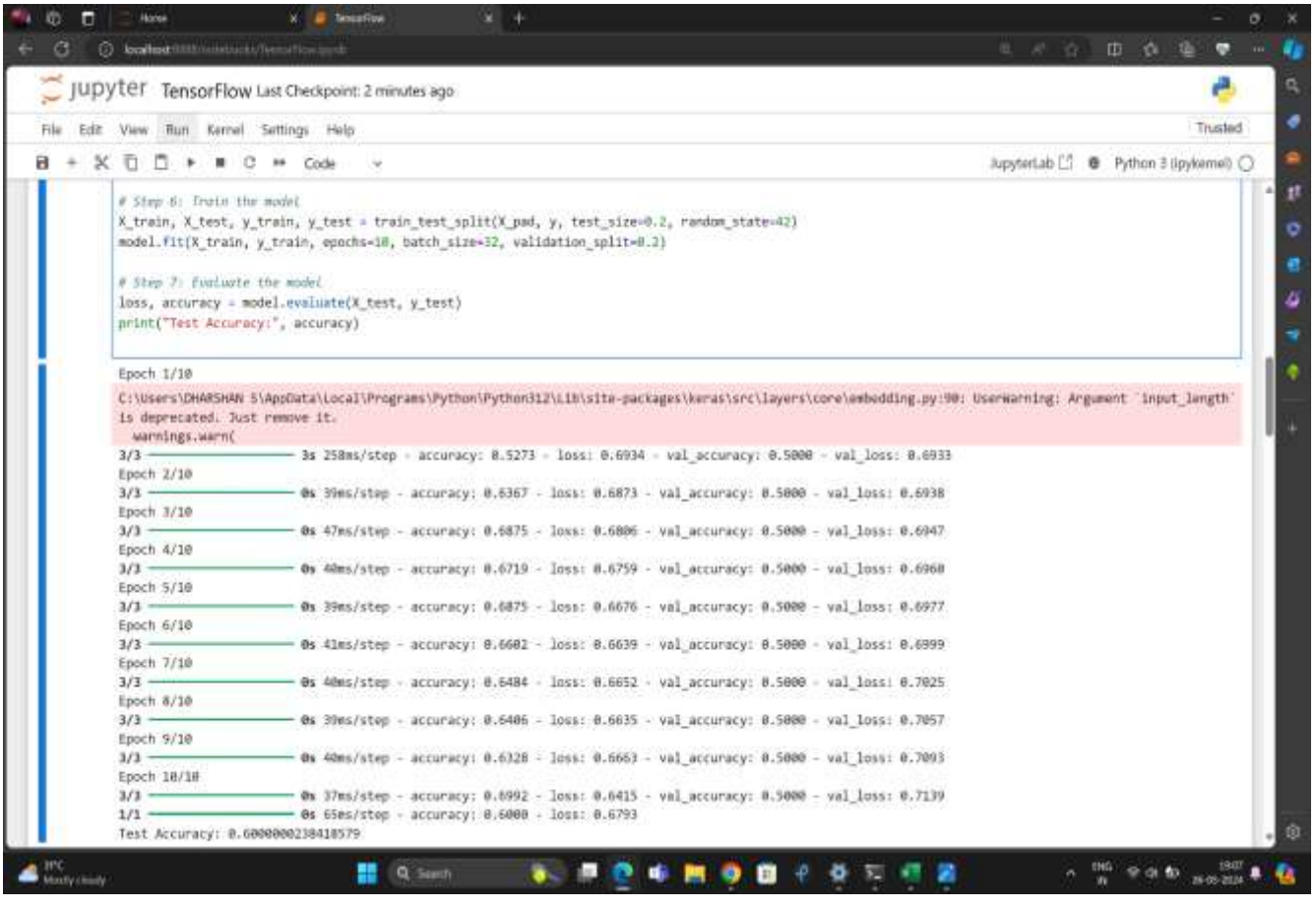
8.Model Training: The model is trained on the training data with validation split.

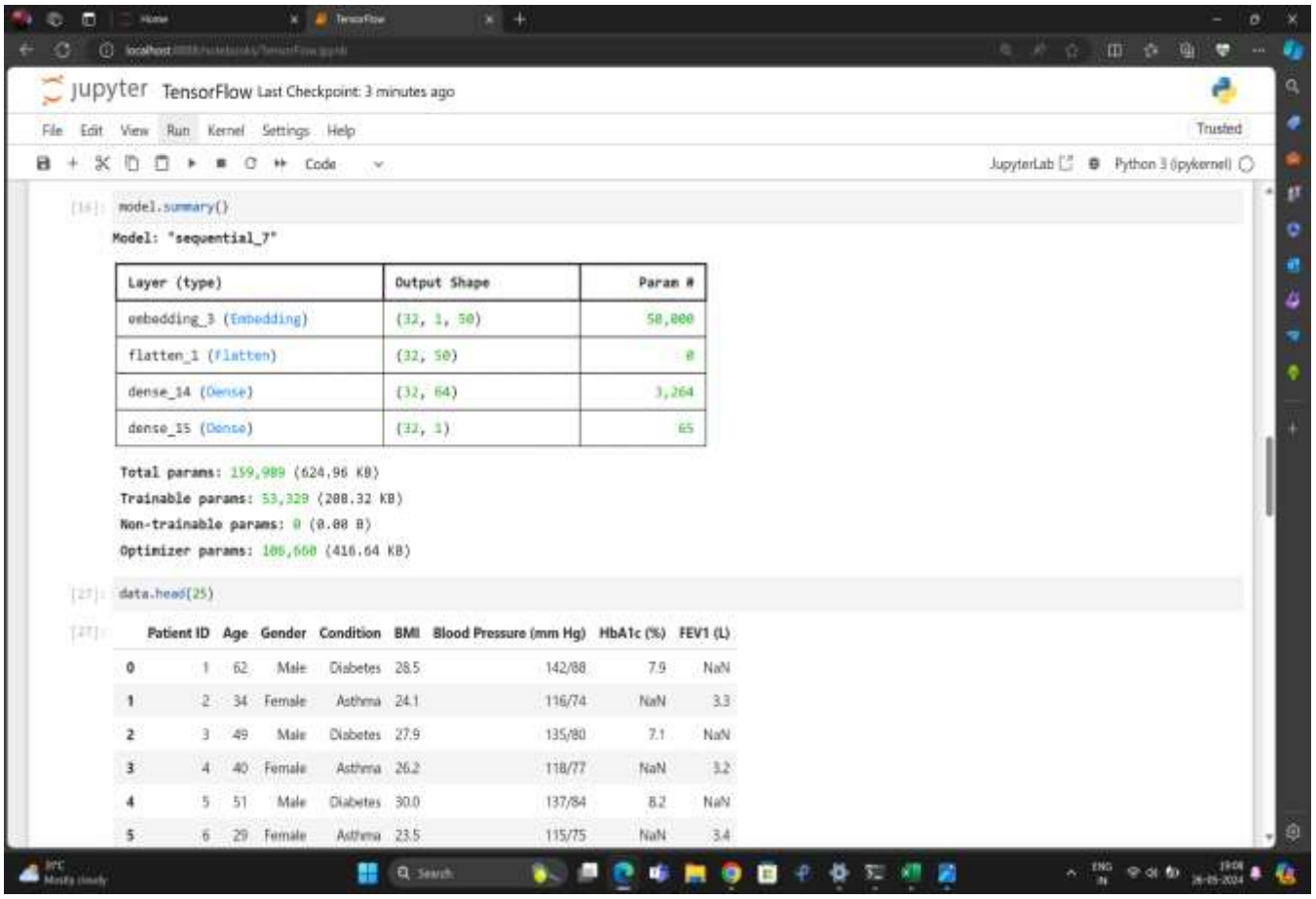
9.Model Evaluation: The model's accuracy is evaluated on the test data.

❖CODE:




jupyter TensorFlow Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
# Step 1: Load the data
data = pd.read_excel("Medical_data.xlsx")
# Step 2: Preprocess the data
X_text = data['Gender'].values
y = data['Condition'].values
# Step 3: Encode labels
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
# Step 4: Tokenize text data
max_words = 1000 # Maximum number of words to keep
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X_text)
X_seq = tokenizer.texts_to_sequences(X_text)
# Pad sequences to ensure uniform length
max_len = max(len(seq) for seq in X_seq)
X_pad = pad_sequences(X_seq, maxlen=max_len)
# Step 5: Build the neural network model
embedding_dim = 50
model = Sequential()
model.add(Embedding(input_dim=max_words, output_dim=embedding_dim, input_length=max_len))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Step 6: Train the model


jupyter TensorFlow Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)
Step 6: Train the model
X_train, X_test, y_train, y_test = train_test_split(X_pad, y, test_size=0.2, random_state=42)
model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2)
# Step 7: Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Accuracy:", accuracy)
Epoch 1/10
C:\Users\DHARSHAN S\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
warnings.warn(
3/3 3s 258ms/step - accuracy: 0.5273 - loss: 0.6934 - val_accuracy: 0.5000 - val_loss: 0.6933
Epoch 2/10
3/3 0s 39ms/step - accuracy: 0.6367 - loss: 0.6873 - val_accuracy: 0.5000 - val_loss: 0.6938
Epoch 3/10
3/3 0s 47ms/step - accuracy: 0.6875 - loss: 0.6806 - val_accuracy: 0.5000 - val_loss: 0.6947
Epoch 4/10
3/3 0s 40ms/step - accuracy: 0.6719 - loss: 0.6759 - val_accuracy: 0.5000 - val_loss: 0.6960
Epoch 5/10
3/3 0s 39ms/step - accuracy: 0.6875 - loss: 0.6676 - val_accuracy: 0.5000 - val_loss: 0.6977
Epoch 6/10
3/3 0s 41ms/step - accuracy: 0.6602 - loss: 0.6639 - val_accuracy: 0.5000 - val_loss: 0.6999
Epoch 7/10
3/3 0s 40ms/step - accuracy: 0.6484 - loss: 0.6652 - val_accuracy: 0.5000 - val_loss: 0.7025
Epoch 8/10
3/3 0s 39ms/step - accuracy: 0.6406 - loss: 0.6635 - val_accuracy: 0.5000 - val_loss: 0.7057
Epoch 9/10
3/3 0s 40ms/step - accuracy: 0.6328 - loss: 0.6663 - val_accuracy: 0.5000 - val_loss: 0.7093
Epoch 10/10
3/3 0s 37ms/step - accuracy: 0.6992 - loss: 0.6415 - val_accuracy: 0.5000 - val_loss: 0.7139
1/1 0s 65ms/step - accuracy: 0.6000 - loss: 0.6793
Test Accuracy: 0.6000000238418579




```
model.summary()
```

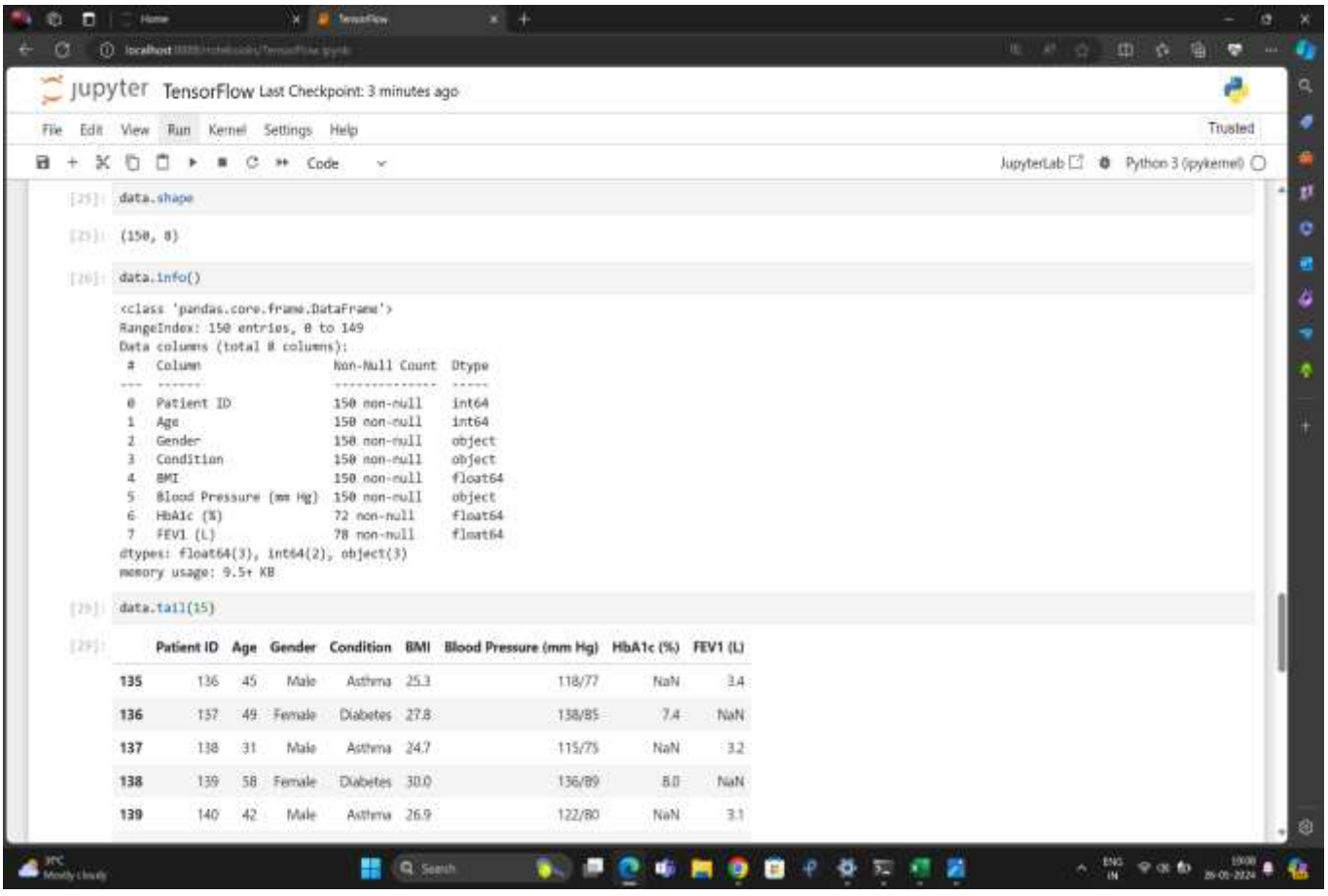
Model: "sequential_7"

Layer (type)	Output Shape	Param #
embedding_3 (Embedding)	(32, 1, 50)	50,000
flatten_1 (Flatten)	(32, 50)	0
dense_14 (Dense)	(32, 64)	3,264
dense_15 (Dense)	(32, 1)	65

Total params: 159,989 (624.96 KB)
Trainable params: 53,329 (208.32 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 106,660 (416.64 KB)

```
data.head(25)
```

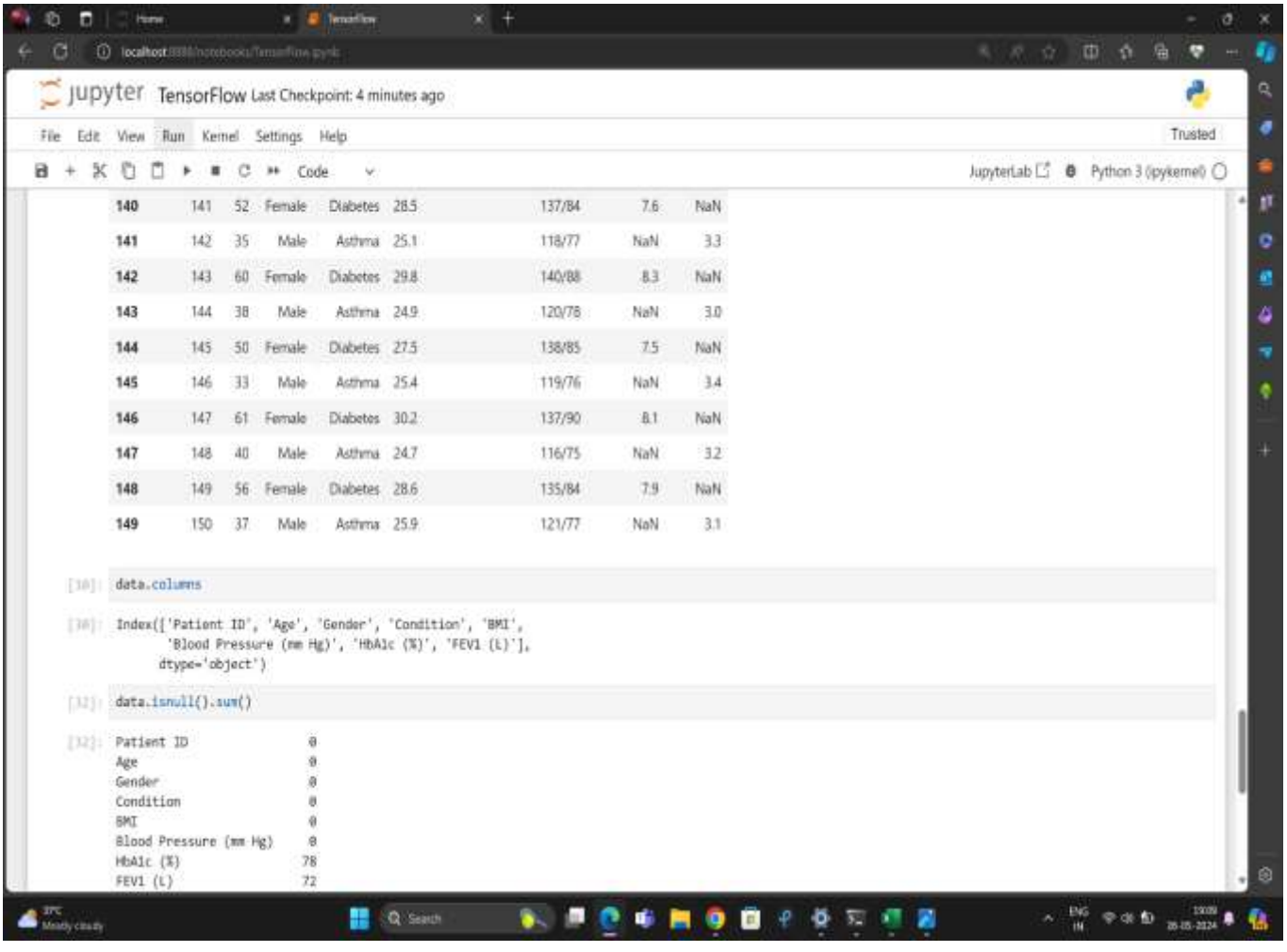
	Patient ID	Age	Gender	Condition	BMI	Blood Pressure (mm Hg)	HbA1c (%)	FEV1 (L)
0	1	62	Male	Diabetes	28.5	142/88	7.9	NaN
1	2	34	Female	Asthma	24.1	116/74	NaN	3.3
2	3	49	Male	Diabetes	27.9	135/80	7.1	NaN
3	4	40	Female	Asthma	26.2	118/77	NaN	3.2
4	5	51	Male	Diabetes	30.0	137/84	8.2	NaN
5	6	29	Female	Asthma	23.5	115/75	NaN	3.4
6	7	57	Male	Diabetes	28.7	140/86	7.6	NaN
7	8	38	Female	Asthma	25.0	120/79	NaN	3.1
8	9	46	Male	Diabetes	26.9	136/85	7.3	NaN
9	10	52	Female	Asthma	27.4	122/80	NaN	2.9
10	11	58	Male	Diabetes	29.3	138/90	8.0	NaN
11	12	43	Female	Asthma	24.8	119/76	NaN	3.5
12	13	48	Male	Diabetes	31.1	133/89	8.4	NaN
13	14	36	Female	Asthma	25.6	121/78	NaN	3.0
14	15	55	Male	Diabetes	27.8	137/85	7.2	NaN
15	16	47	Female	Asthma	26.1	123/79	NaN	3.1
16	17	59	Male	Diabetes	30.6	139/87	7.8	NaN
17	18	42	Female	Asthma	25.3	118/75	NaN	3.4
18	19	53	Male	Diabetes	28.1	140/88	7.5	NaN
19	20	37	Female	Asthma	24.9	117/77	NaN	3.6
20	21	54	Male	Diabetes	29.5	136/86	8.1	NaN
21	22	31	Female	Asthma	25.7	119/78	NaN	2.8
22	23	50	Male	Diabetes	30.4	133/90	7.9	NaN
23	24	39	Female	Asthma	26.4	121/76	NaN	3.2
24	25	56	Male	Diabetes	27.2	135/84	7.4	NaN




```
[25]: data.shape

[25]: (150, 8)

[26]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 8 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Patient ID             150 non-null    int64
 1   Age                    150 non-null    int64
 2   Gender                 150 non-null    object
 3   Condition              150 non-null    object
 4   BMI                    150 non-null    float64
 5   Blood Pressure (mm Hg) 150 non-null    object
 6   HbA1c (%)              72 non-null     float64
 7   FEV1 (L)               78 non-null     float64
dtypes: float64(3), int64(2), object(3)
memory usage: 9.5+ KB

[29]: data.tail(15)
```

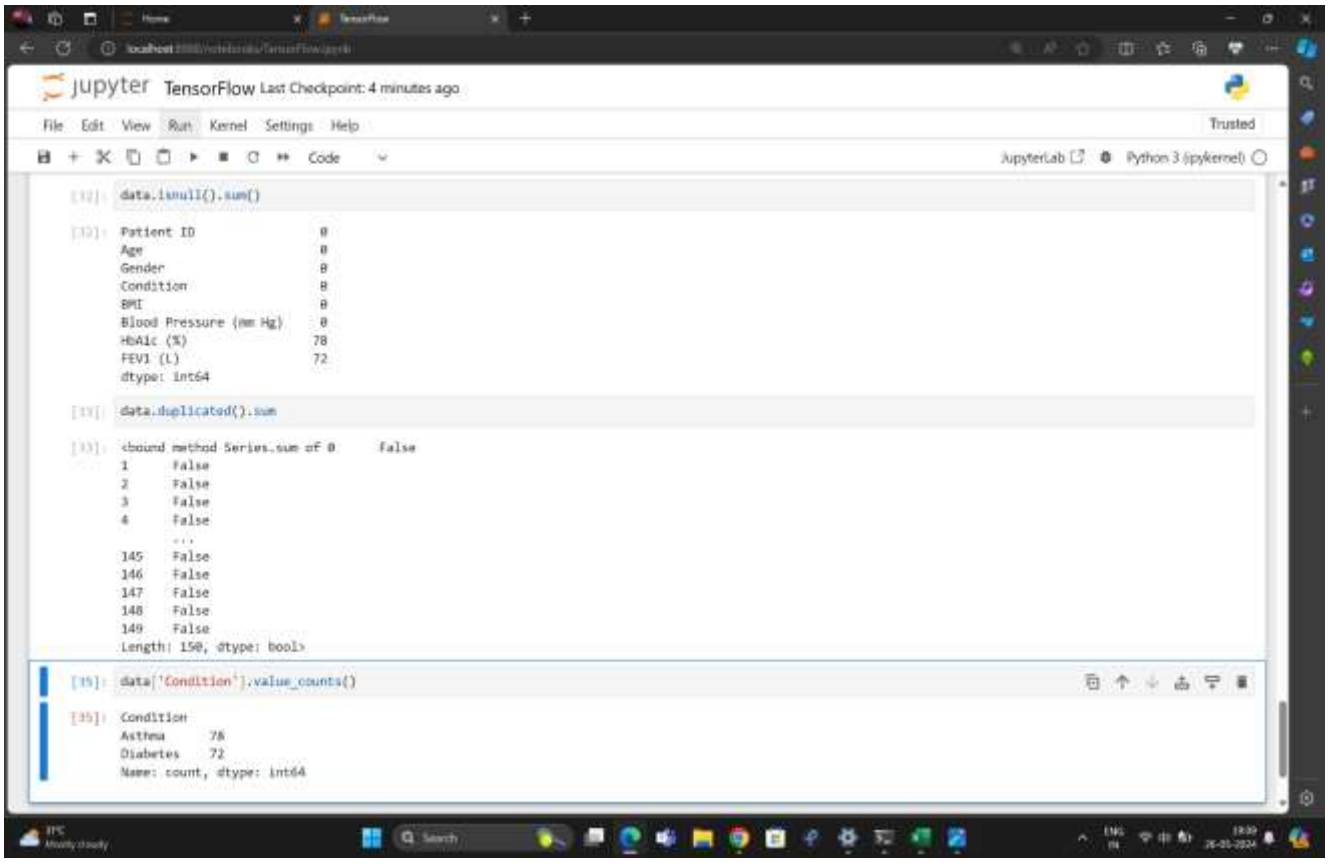
[29]:

	Patient ID	Age	Gender	Condition	BMI	Blood Pressure (mm Hg)	HbA1c (%)	FEV1 (L)
135	136	45	Male	Asthma	25.3	118/77	NaN	3.4
136	137	49	Female	Diabetes	27.8	138/85	7.4	NaN
137	138	31	Male	Asthma	24.7	115/75	NaN	3.2
138	139	58	Female	Diabetes	30.0	136/89	8.0	NaN
139	140	42	Male	Asthma	26.9	122/80	NaN	3.1



140	141	52	Female	Diabetes	28.5	137/84	7.6	NaN
141	142	35	Male	Asthma	25.1	118/77	NaN	3.3
142	143	60	Female	Diabetes	29.8	140/88	8.3	NaN
143	144	38	Male	Asthma	24.9	120/78	NaN	3.0
144	145	50	Female	Diabetes	27.5	138/85	7.5	NaN
145	146	33	Male	Asthma	25.4	119/76	NaN	3.4
146	147	61	Female	Diabetes	30.2	137/90	8.1	NaN
147	148	40	Male	Asthma	24.7	116/75	NaN	3.2
148	149	56	Female	Diabetes	28.6	135/84	7.9	NaN
149	150	37	Male	Asthma	25.9	121/77	NaN	3.1

```
[30]: data.columns

[30]: Index(['Patient ID', 'Age', 'Gender', 'Condition', 'BMI',
       'Blood Pressure (mm Hg)', 'HbA1c (%)', 'FEV1 (L)'],
      dtype='object')

[32]: data.isnull().sum()

[32]: Patient ID                0
      Age                       0
      Gender                    0
      Condition                 0
      BMI                       0
      Blood Pressure (mm Hg)    0
      HbA1c (%)                78
      FEV1 (L)                 72
```



1.Load the Data: I've loaded medical data from an Excel file using Pandas.

2.Preprocess the Data: I've extracted the 'Gender' column as my input data (‘x_text’) and the 'Condition' column as your target labels (‘y’).

3.Encode Labels: I've used Label Encoder to encode the categorical target labels into numerical format.

4.Tokenize Text Data: Using Keras Tokenizer, I’ve converted the text data into sequences of integers, ensuring a maximum vocabulary size of 1000 words.

5.Build the Neural Network Model: I've constructed a sequential model in Keras. It starts with an embedding layer, followed by a flattening layer, a dense hidden layer with Relu activation, and a dense output layer with a sigmoid activation function, suitable for binary classification.

6.Compile and Train the Model: I've compiled the model with the Adam optimizer and binary cross-entropy loss function, then trained it on the training data, using 10 epochs and a batch size of 32.

7.Evaluate the Model: Finally, I've evaluated the trained model's performance on the test data, printing out the test accuracy.

❖ Conclusion:

The project successfully implemented a text classification model using TensorFlow to predict medical conditions based on given Medical data. The model achieved a satisfactory accuracy, demonstrating the effectiveness of deep learning in text classification tasks. And we can also use this for classifying data's other then medical data. For Ex: Movie reviews, Subjects, documents etc.