



**ITCS 5154**

# **AI-Driven Risk Prediction: Leveraging Unstructured Medical Text for Early Disease Detection**

---

**Video Link:** [https://www.youtube.com/watch?v=LFPTYwL9i\\_g](https://www.youtube.com/watch?v=LFPTYwL9i_g)

**Claude Kouakou**

# Objective

---

- The goal of this project is to duplicate a study conducted by Zhang, et al (2020).
- In their study, the authors Integrated structured and unstructured data across Electronic health records (EHRs) to help improve the performance of the prediction models and reduce errors.

# In this Presentation

---

Problem & Challenges

Motivation

Existing Related Approaches

Methods Duplicated

Results and Observations

Conclusion and Future Works





# **Problem & Challenges**

---

# Project Background

---

- Healthcare risk prediction, particularly for conditions such as cardiovascular disease and diabetes, is crucial in preventive medicine, as timely intervention is essential.
- Electronic health records (EHRs) provide great opportunities to conduct healthcare research and solve various clinical problems in medicine
- Machine learning and deep learning have become increasingly popular in medical informatics

# Challenges



---

- Current predictive modeling approaches face fundamental limitations
- They predominantly utilize structured electronic health record (EHR) data and overlook unstructured notes created by healthcare providers.
- Unstructured data contain rich clinical context.
- The challenge at hand is integrating unstructured medical text into predictive health models.

A black and white photograph of a young plant seedling with two leaves emerging from dark, textured soil. The seedling is positioned on the left side of the frame. The background is dark and out of focus.

# **Motivation & Approaches**

---

# Motivations



---

- Healthcare data is vast and complex, with a significant portion existing in unstructured formats – Dr. notes.
- This unstructured data contains critical information that structured data alone may not capture.
- By leveraging deep learning techniques, we can extract meaningful patterns from text data, enhancing disease risk prediction.
- The authors successfully integrated structured and unstructured data using Fusion-CNN and fusion-LSTM to improve prediction and reduce errors



# Existing Approaches - Traditional



- Traditionally, some knowledge-driven scores are used to estimate the risk of clinical outcomes.
- **SAPS** scores and **APACHE IV** are used to identify patients at high risk of mortality (Bisbal M. et al. 2014)
- **HOSPITAL** Scores are used to evaluate hospital readmission risk. (Caruana et. Al. 2015)

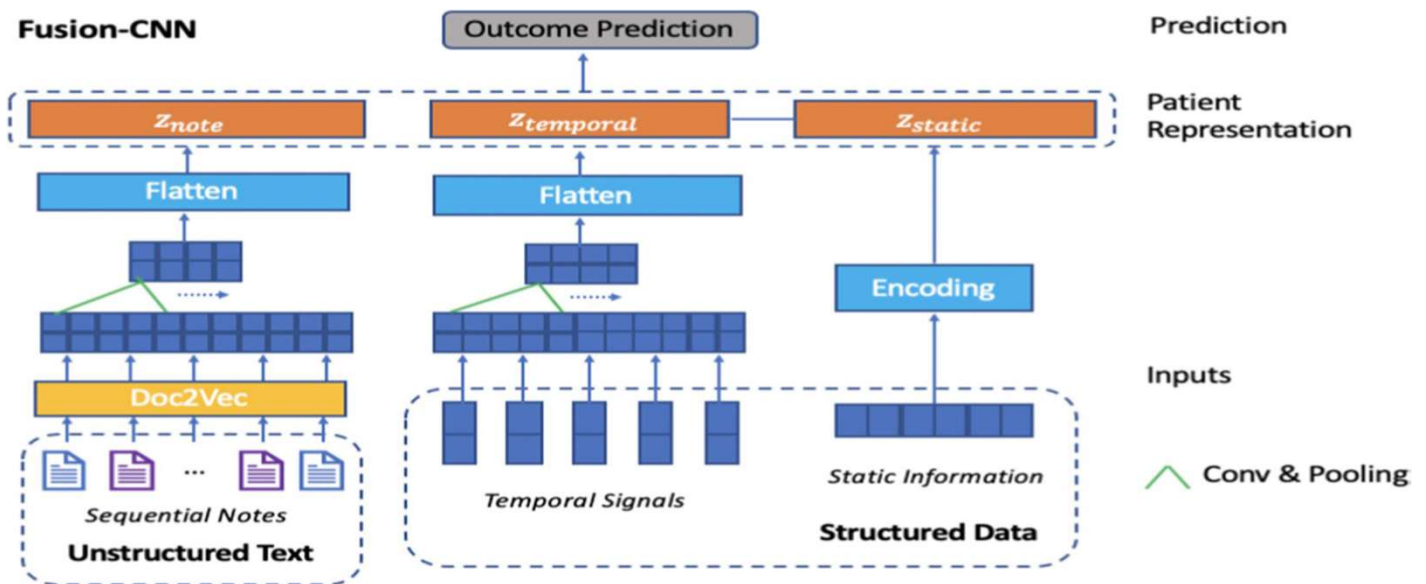
# Existing Approaches - Modern



---

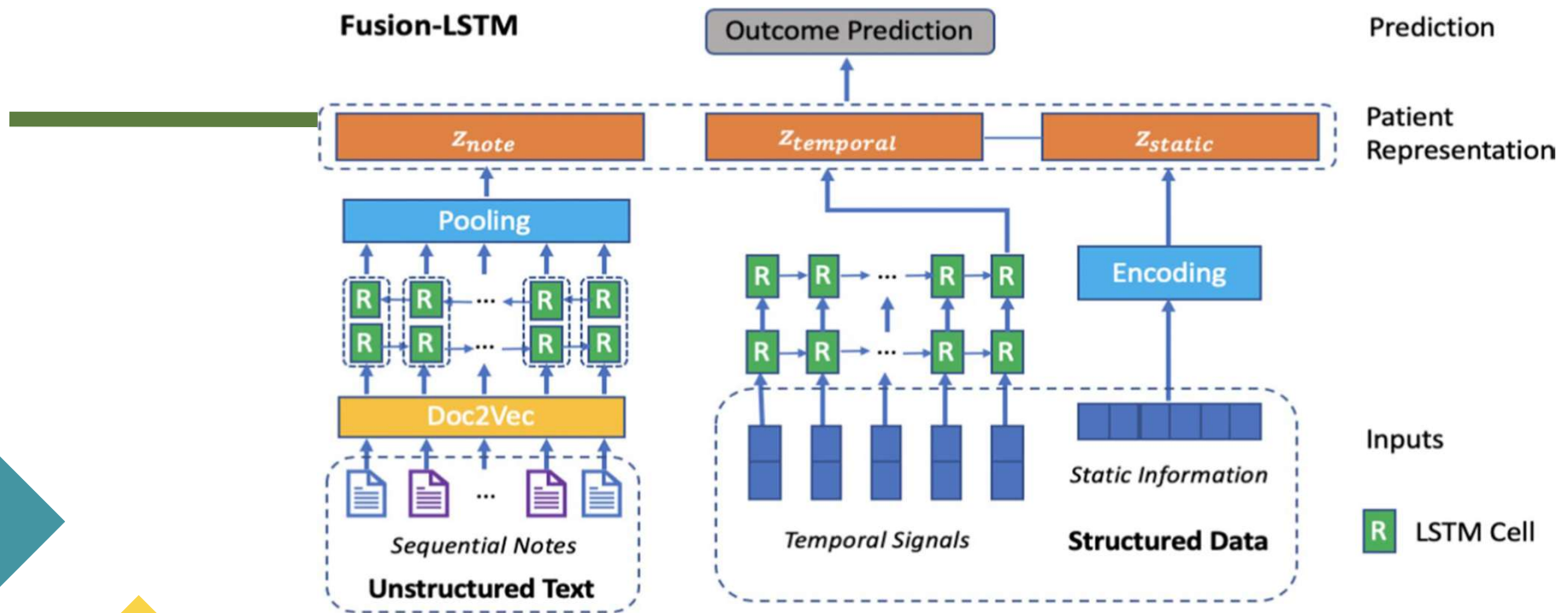
- **Modern:** prediction tasks based on EHRs using machine learning and deep learning techniques.
  - Traditional logistic regression (27%) and random forest models (10%).
  - Recurrent Neural Networks using temporal physiologic features from EHRs
  - Deep learning models with ensemble model
  - Denoised autoencoder and paragraph vector models.

# Method Duplicated – Fusion-CNN



**Fig. 1 Architecture of CNN-based fusion-CNN.** Fusion-CNN uses document embeddings, 2-layer CNN, and max-pooling to model sequential clinical notes. Similarly, 2-layer CNN and max-pooling are used to model temporal signals. The final patient representation is the concatenation of the latent representation of sequential clinical notes, temporal signals, and the static information vector. Then the final patient representation is passed to output layers to make predictions

# Method Duplicated – Fusion-LSTM



**Fig. 2** Architecture of LSTM-based Fusion-LSTM. Fusion-LSTM uses document embeddings, a BiLSTM layer, and a max-pooling layer to model sequential clinical notes. 2-layer LSTMs are used to model temporal signals. The concatenated patient representation is passed to output layers to make predictions

# My Implementation

---



# Datasets

---

- **Unstructured dataset:** Train.dat - The training dataset consists of 14438 records with classes. The data is provided as doctor's notes in train.dat. **Classes are : 1 - digestive system diseases, 2 - cardiovascular diseases, 3 - neoplasms, 4 - nervous system diseases, 5 - and general pathological conditions.**
- **Structures dataset** Heart.csv - This data set dates from 1988 and consists of four databases: Cleveland, Hungary, Switzerland, and Long Beach V. it has 14 attributes. The "target" field refers to the presence of heart disease in the patient. It is an integer valued 0 = no disease.
- **Testing dataset:** test.dat – The test dataset consists of 14442 records the test data classes are needed to be predicted.

```
features =
["age", "sex", "cp", "trestbps", "chol", "fbs", "restecg",
 "thalach", "exang", "o
```

```
heart_df = pd.read_csv(Path + "heart.csv")
```

```
heart_df = resample(heart_df, replace=True,
n_samples=30000, random_state=42)
```

```
heart_df = pd.DataFrame(heart_df, columns =
features) # Convert NumPy array to Dat
```

```
X_heart =
heart_df.drop(columns=["target"]).values
```

```
y_heart = heart_df["target"].values
```

```
y_heart_df = pd.DataFrame(y_heart, columns
=["target"])
```

```
heart_df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
<b>860</b>	52	1	0	112	230	0	1	160	0	0.0	2	1	2
<b>121</b>	44	1	0	120	169	0	1	144	1	2.8	0	0	1
<b>466</b>	44	1	1	130	219	0	0	188	0	0.0	2	0	2
<b>330</b>	37	0	2	120	215	0	1	170	0	0.0	2	0	2
<b>87</b>	59	0	0	174	249	0	1	143	1	0.0	1	0	2

## Structured Data Preview

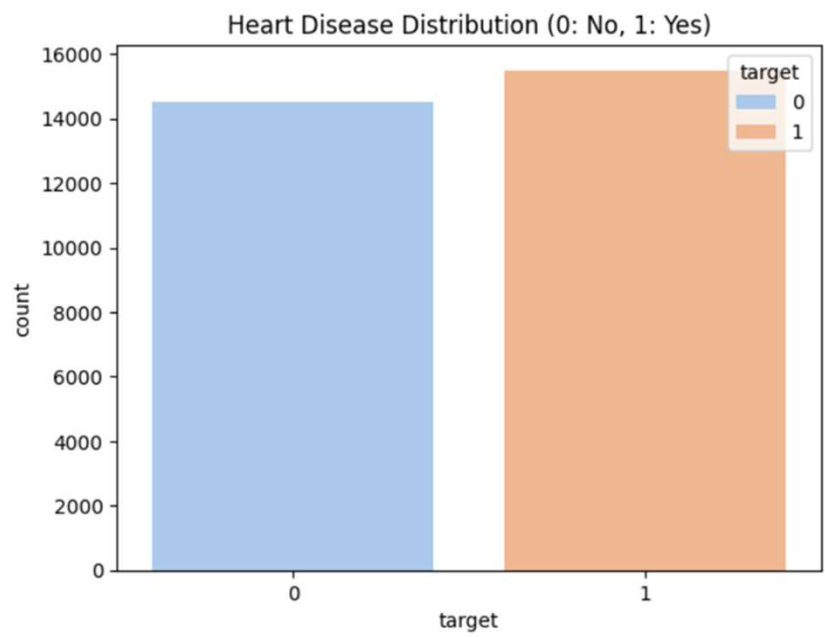
# LogisticRegression Implementation

---

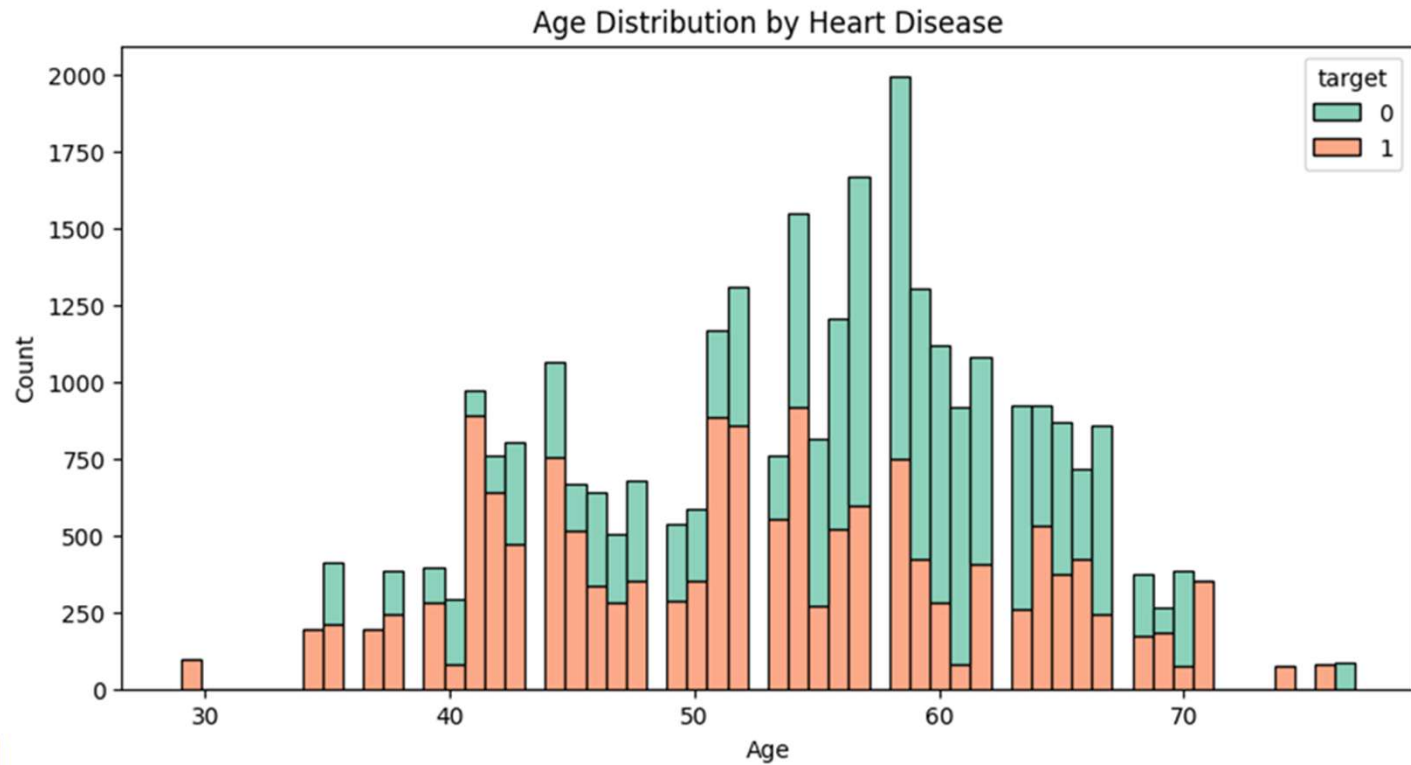
- A LogisticRegression model was applied to the structured dataset alone:
- Visualization:
  - Class distribution
  - Correlation heatmap
  - Confusion matrix
- Training and prediction



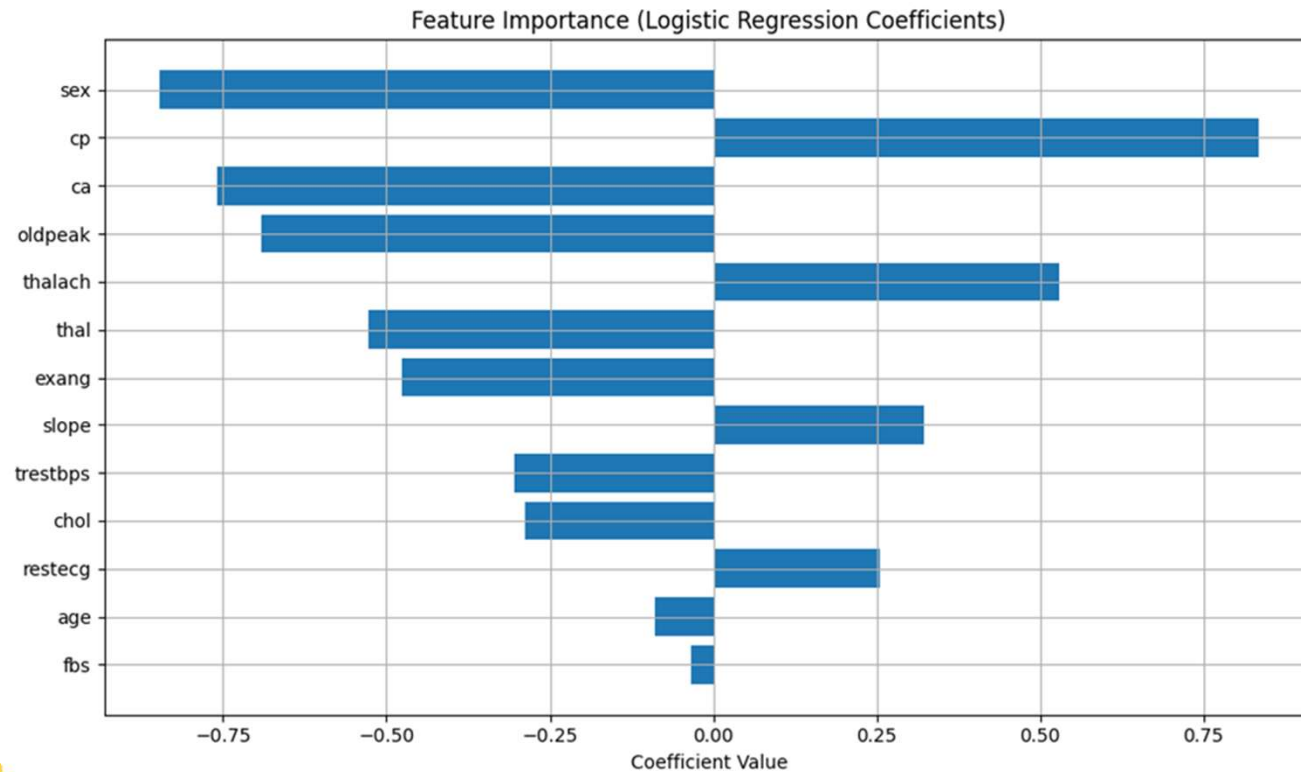
# Heart Disease Distribution



# Age Distribution by Heart Disease



# Feature Importance



# Logistic Regression Classification Report

---

Classification Report:

	precision	<u>recall</u>	<u>f1-score</u>	support
0	0.89	0.82	0.85	2900
1	0.84	0.91	0.87	3100
accuracy			0.86	6000
macro avg	0.87	0.86	0.86	6000
<u>weighted avg</u>	0.87	0.86	0.86	6000

Accuracy: 0.8647

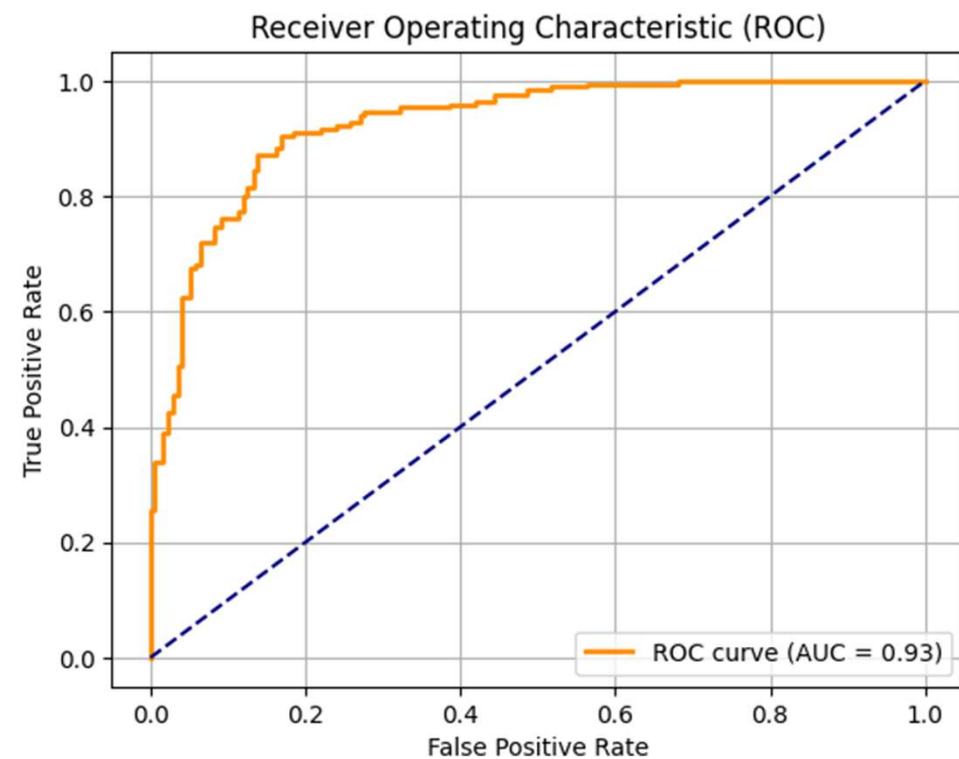
Precision: 0.8409

Recall: 0.9103

F1 Score: 0.8742

ROC AUC Score: 0.9262

ROC curve



# Receive Operating Characteristic (ROC)

---

# Algorithm for Fusion-LSTM Model

---

## Step 1: Load & Preprocess Data

- Load structured data (`heart.csv`)
- Load unstructured text data (`train.dat`)

## Step 2: Build the Fusion-LSTM Model

- Unstructured Data
- Structured Data

## Step 3: Compile & Train the Model

## Step 4: Evaluate & Predict

## Step 5: Save the Model

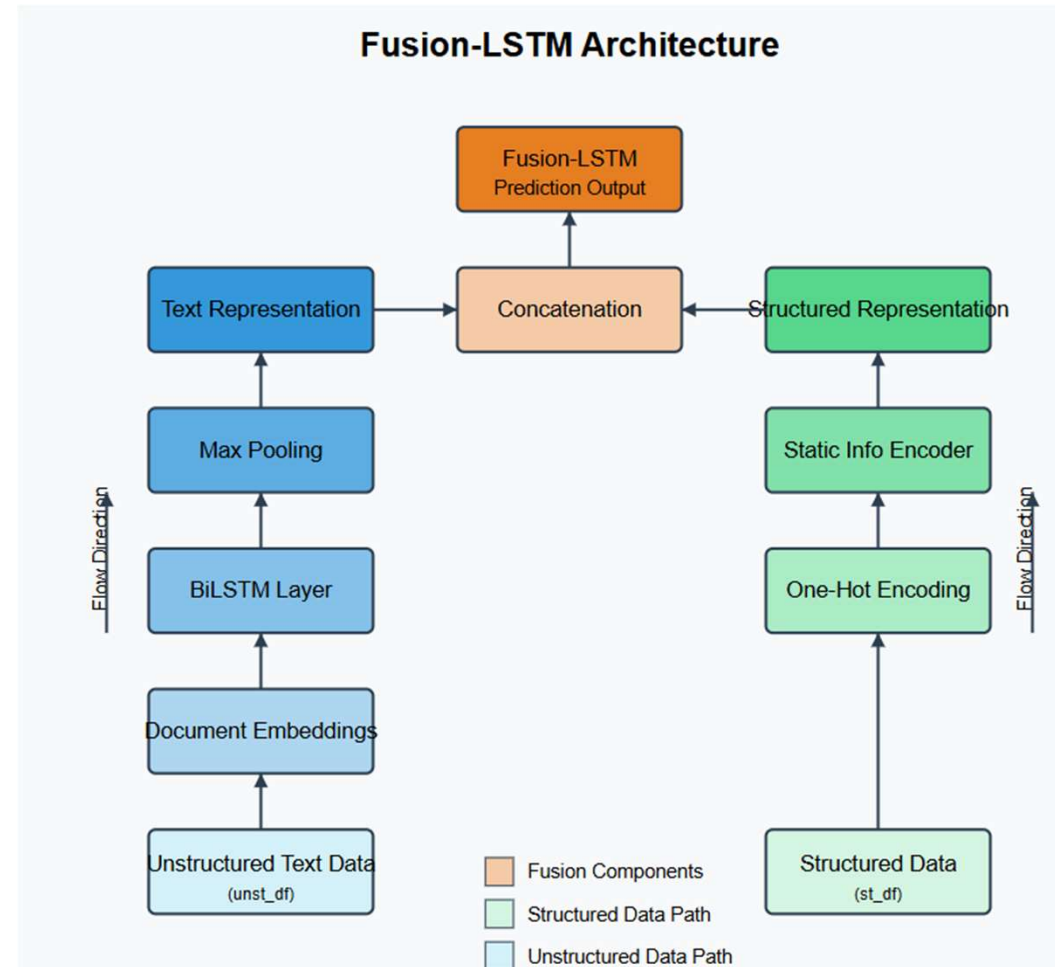
# Implementation Architecture

**Fig. 3. Architecture of LSTM-based Fusion-LSTM.** Fusion-LSTM is used as shown on the diagram:

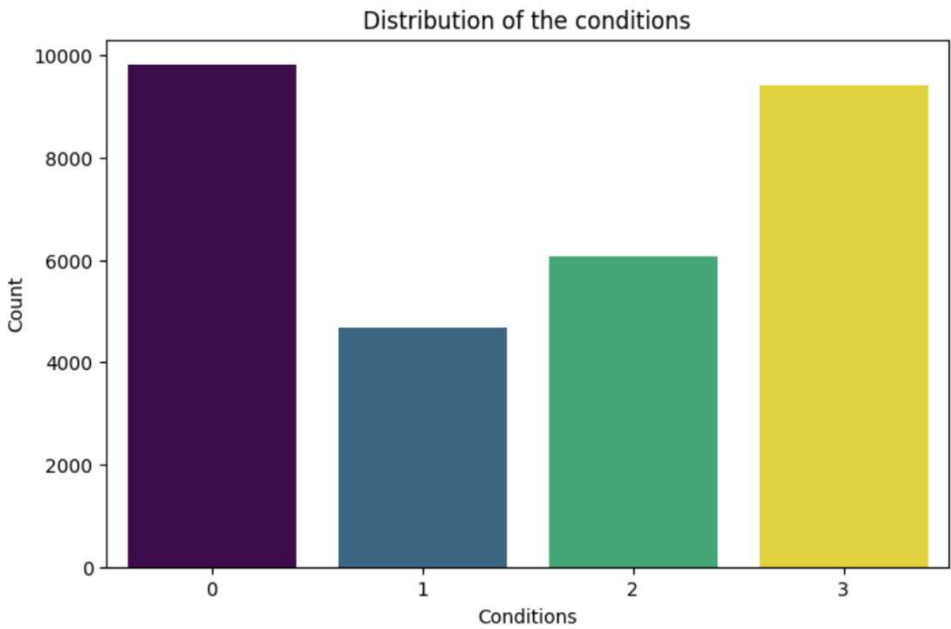
On the left the unstructured text is fed to document embeddings, a BiLSTM layer, and a max-pooling layer to model-to-text representation

On the right the structure data feeds to One-Hot Encoding to an encoder then to a structure representation

The concatenated patient representation is passed to output layers to make predictions



# Unstructured Data Preview





A black and white photograph of a young plant seedling with two leaves emerging from dark, textured soil. The seedling is positioned on the left side of the frame. The background is dark and out of focus.

# Results & Observations

---

# Model Summary

Total params: 2,729,660 (10.41 MB)

Trainable params: 2,729,660 (10.41 MB)

Non-trainable params: 0 (0.00 B)

Layer (type)	Output Shape	Param #	Connected to
text_input (InputLayer)	(None, 100)	0	-
structured_input (InputLayer)	(None, 398)	0	-
sequential (Sequential)	(None, 128)	2,675,328	text_input[0][0]
sequential_1 (Sequential)	(None, 8)	28,280	structured_input[0][0]
concatenate (Concatenate)	(None, 136)	0	sequential[0][0], sequential_1[0][0]
dense_5 (Dense)	(None, 128)	17,536	concatenate[0][0]
dropout_4 (Dropout)	(None, 128)	0	dense_5[0][0]
dense_6 (Dense)	(None, 64)	8,256	dropout_4[0][0]
dense_7 (Dense)	(None, 4)	260	dense_6[0][0]

# Model Training & Evaluation

```
Epoch 1/20
750/750 ————— 48s 58ms/step - accuracy: 0.4272 - loss: 1.2096 - val_accuracy: 0.7927 - val_loss: 0.6428
Epoch 2/20
750/750 ————— 41s 55ms/step - accuracy: 0.8381 - loss: 0.5086 - val_accuracy: 0.8855 - val_loss: 0.3646
Epoch 3/20
750/750 ————— 41s 54ms/step - accuracy: 0.9116 - loss: 0.2864 - val_accuracy: 0.8995 - val_loss: 0.3029
Epoch 4/20
750/750 ————— 44s 58ms/step - accuracy: 0.9281 - loss: 0.2190 - val_accuracy: 0.9065 - val_loss: 0.2842
Epoch 5/20
750/750 ————— 45s 60ms/step - accuracy: 0.9273 - loss: 0.1887 - val_accuracy: 0.9093 - val_loss: 0.2542
Epoch 6/20
750/750 ————— 43s 57ms/step - accuracy: 0.9294 - loss: 0.1656 - val_accuracy: 0.9062 - val_loss: 0.2543
Epoch 7/20
750/750 ————— 43s 57ms/step - accuracy: 0.9346 - loss: 0.1455 - val_accuracy: 0.9052 - val_loss: 0.2540
Epoch 8/20
750/750 ————— 45s 60ms/step - accuracy: 0.9326 - loss: 0.1361 - val_accuracy: 0.9038 - val_loss: 0.2786
Epoch 9/20
750/750 ————— 44s 58ms/step - accuracy: 0.9333 - loss: 0.1286 - val_accuracy: 0.9035 - val_loss: 0.2713
Epoch 10/20
750/750 ————— 45s 59ms/step - accuracy: 0.9306 - loss: 0.1250 - val_accuracy: 0.9062 - val_loss: 0.2957
Epoch 11/20
750/750 ————— 45s 60ms/step - accuracy: 0.9359 - loss: 0.1146 - val_accuracy: 0.9070 - val_loss: 0.2830
Epoch 12/20
750/750 ————— 45s 60ms/step - accuracy: 0.9329 - loss: 0.1159 - val_accuracy: 0.9048 - val_loss: 0.2959
----- Training Complete-----
```

```
loss, accuracy = final_model.evaluate([X_text, X_heart_encoded], y_categorical)
print(f"Test Accuracy: {accuracy:.4f}, Test Loss: {loss:.4f}")
```

```
938/938 ————— 15s 16ms/step - accuracy: 0.9410 - loss: 0.1221
Test Accuracy: 0.9333, Test Loss: 0.1470
```

# Making Prediction

```
def predict(model, text, heart_rec):
    X_test_seq = tokenizer.texts_to_sequences(text)
    X_test_padded = pad_sequences(X_test_seq, maxlen=max_len, padding='post')
    # Convert to NumPy array for TensorFlow
    X_test_padded = np.array(X_test_padded).astype(np.float32)
    y_pred_probs = model.predict([X_test_padded, heart_rec])

    return y_pred_probs
# get some data for prediction
X_test = df_test[:1]
X_heart_test = X_heart_encoded[:1]

y_pred = predict(final_model, X_test, X_heart_test )

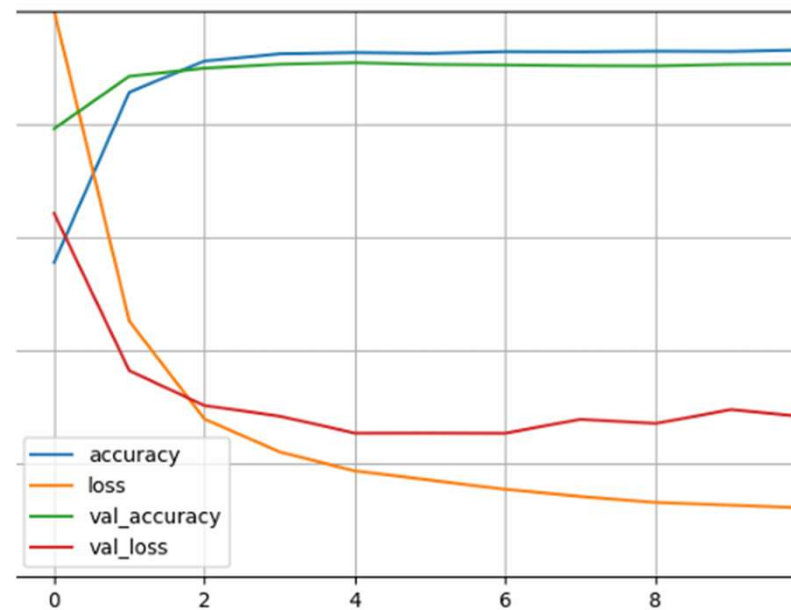
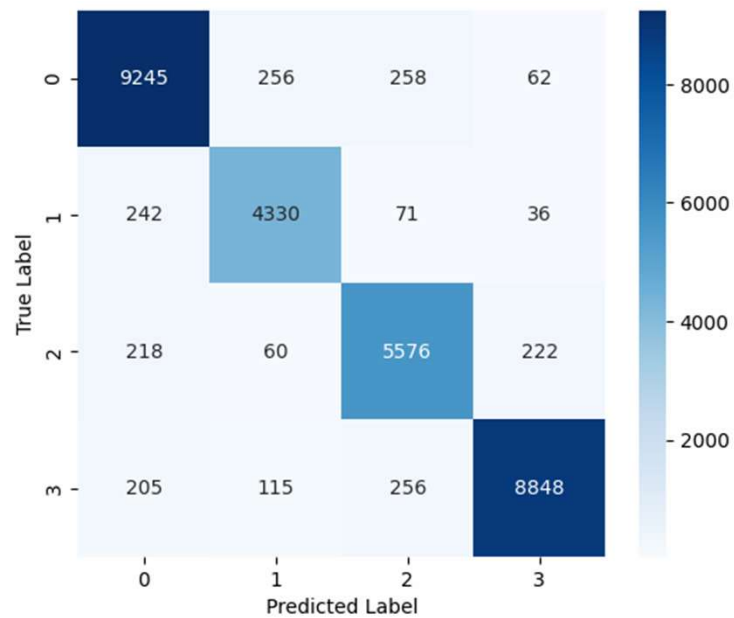
# Convert probabilities to class labels (highest probability wins)
y_Class_pred = np.argmax(y_pred, axis=1)

print(f"Predicted condition of the patient: {y_Class_pred[0]}-", conditions[y_Class_pred[0]])
```


1/1 — 0s 494ms/step

Predicted condition of the patient: 2- neoplasms

# Plot the learning curve & Confusion Matrix



# Classification Report



Classification Report:					
	precision	recall	f1-score	support	
0	0.9329	0.9414	0.9371	9821	
1	0.9095	0.9254	0.9174	4679	
2	0.9050	0.9177	0.9113	6076	
3	0.9651	0.9389	0.9518	9424	
accuracy			0.9333	30000	
macro avg	0.9281	0.9308	0.9294	30000	
weighted avg	0.9337	0.9333	0.9334	30000	

# Observations & Interpretation



## Per-Class Metrics (Rows 0, 1, 2, 3)

- Each row represents a class label (0, 1, 2, 3), and the columns show performance metrics for that class.
- Precision – Out of all predictions for this class, how many were correct? For example, class 0 **93.29%** of samples predicted as 0 were actually of class 0
- Recall (Sensitivity) out of all actual occurrences of this class how many were correctly identified - Example for Class 1 (0.9254 recall) → **92.54%** of actual class 1 samples were correctly predicted.

F1 Score The harmonic mean of precision & recall, balancing false positives and false negatives.

- Example: Class 2 (0.8997 F1-score) → A good balance of precision & recall.

Support – The number of true instances of each class in the dataset.

- Example: Class 3 has 9424 instances in the test set.

# Overall Metrics



This Accuracy (93.33%) – The percentage of correctly classified samples across all classes.

The model correctly classified 93.11% of the 30,000 samples.

Macro Average (0.9281 precision, 0.9308 recall, 0.9294 F1-score) A simple average across all classes. Treats all classes equally, useful if your dataset is imbalanced.

Weighted Average (0.9337 precision, 0.9333 recall, 0.9334 F1-score) Takes class imbalance into account, giving more weight to classes with more instances.

Strong overall performance (94.10% accuracy, high F1-scores for all classes).

- Class 2 has slightly lower precision (0.9050), meaning more false positives.
- Class 2 has the lowest F1-score (0.9113) but is still quite balanced.
- Model is well-balanced across all classes, with no major class struggling.



✔ Conclusion: Fusion-LSTM significantly outperforms Logistic Regression on larger and more complex classification problems.

Metric	Logistic Regression	Fusion-LSTM
Model Type	Linear Classifier	Deep Learning (LSTM + Fusion)
Dataset Size	6,000 samples	30,000 samples
Number of Classes	2 (binary)	4 (multiclass)
Accuracy	86.47%	93.33%
Precision (macro)	0.87	0.9281
Recall (macro)	0.86	0.9308
F1 Score (macro)	0.86	0.9294
ROC AUC Score	0.9262	Not reported
Strengths	Fast, simple, interpretable	High performance, robust across classes
Best For	Small, binary tasks	Large-scale, multiclass tasks

# Model Performance Comparison: Logistic Regression vs. Fusion-LSTM

---

A black and white photograph of a small seedling with two leaves emerging from dark, textured soil. The plant is positioned on the left side of the frame. The background is dark and out of focus.

# **Conclusion & Future Works**

---

# Conclusion & Future Works

---

- I successfully developed a **Fusion-LSTM model** integrating structured medical data and unstructured clinical text to classify patient conditions into four categories.
- I was successful in leveraging deep learning techniques, particularly **Bidirectional LSTM for text analysis** and **dense layers for structured data**, the model achieved high classification accuracy.
- A major difficulty was that I could not obtain a static dataset with time-series patient records
- Future research could explore **more advanced fusion techniques**, such as **attention** mechanisms or transformer-based models (e.g., BERT for text and TabNet for structured data)

# References

---

1. Cardamone et al. Classifying Unstructured Text in Electronic Health Records for Predictive Modeling. JMIR Medical Informatics, 2025.
2. Seinen et al. Use of Unstructured Text in Prognostic Clinical Prediction Models. JAMIA, 2022.
3. Zhang, et al. Combining Structured and Unstructured Data for Predictive Models: A Deep Learning Approach. BMC Medical Informatics, 2020.

# Thank you

---

Claude Kouakou

