SR UNIVERSITY

Project Report on Course

DATA ANALYSIS USING PYTHON (21CS120)

Bachelor of Technology
In

Computer Science & Artificial Intelligence

By

Name: ARIKALA HARSHA VARDHAN **Roll Number: 2203A52002**

Under the Guidance of

Dr. DADI RAMESH
Asst. Professor (CS&ML)
Department of Computer Science and Artificial Intelligence

COMPUTER SCIENCE
SCHOOL OF COMPUTER SCIENCE
AND ARTIFICIAL INTELLIGENCE

SR UNIVERSITY, ANANTHASAGAR, WARANGAL
April, 2025.

SCHOOL OF COMPUTER SCIENCE & ARTIFICIAL INTELLIGENCE

CERTIFICATE OF COMPLETION

This is to certify that **ARIKALA HARSHA VARDHAN** bearing Hall Ticket Number **2203A52002**, a student of **CSE-AIML, 3rd Year - 2nd Semester**, has successfully completed the **Data Analysis Using Python** Course and has submitted the following 3 projects as part of the curriculum:

Project Submissions:

- **CSV** Project**: DIABITES PREDICTION**
- **IMAGE** Project**: Fashion MNIST**
- **TEXT** Project**: DATA FOR TEXT ANALYSIS**

Dr. Dadi Ramesh
Asst. Professor (CSE-AIML)
SR University, Ananthasagar,
Warangal

Date of Completion: 25/04/2025

1) CSV PROJECT: DIABITES PREDICTION

About the Dataset

The **Pima Indians Diabetes Dataset** is a widely used dataset in the medical and machine learning communities. It contains **768 records** of **female Pima Indian patients**, aged 21 years and older, and is used for predicting the onset of **diabetes**. The target variable Outcome is binary:

- 1: Diabetic
- 0: Non-Diabetic

The dataset includes **8 input features** representing physiological and medical test results.

Features of the Dataset

Input Features:

1. **Pregnancies** – Number of pregnancies
2. **Glucose** – Plasma glucose concentration
3. **Blood Pressure** – Diastolic blood pressure
4. **Skin Thickness** – Triceps skinfold thickness
5. **Insulin** – 2-Hour serum insulin
6. **BMI** – Body Mass Index
7. **DiabetesPedigreeFunction** – Diabetes risk based on family history
8. **Age** – Age in years

Target Variable:

- **Outcome** – Indicates diabetes status (0 = Non-Diabetic, 1 = Diabetic)

📊 **Additional Characteristics:**

- All features are **numerical**
- Contains **no categorical or time-based fields**
- Features like **Insulin, SkinThickness, Glucose** may contain **invalid zero values**
- The dataset exhibits **class imbalance** (more non-diabetic cases)
- Some features contain **outliers**

Preprocessing Techniques

1. Handling Missing Values

- Although the dataset has no explicit missing values, certain features (e.g., Glucose, Insulin) contain **zero values** that are **not realistic**.
- These are treated as missing and typically replaced with **median values**.

2. Encoding Categorical Data

- The dataset contains **only numerical data**, so **no encoding** is required.
- The Outcome variable is already binary-encoded.

3. Feature Scaling

- The features have **different ranges** (e.g., Insulin can go up to 846, while Age is typically below 80).
- Scaling ensures that each feature contributes **equally** to the model's learning process.
- **Standardization (Z-score scaling)** is commonly used, transforming features to have a **mean of 0 and standard deviation of 1**.

4. Time-Based Features

- There are **no time-related** fields, so **no temporal preprocessing** is needed.

5. Outlier Detection

- Outliers exist in features like **Insulin and SkinThickness**.
- These can be detected using **boxplots** or statistical methods (e.g., IQR, Z-score), and either removed or treated.

6. Filtering and Cleaning

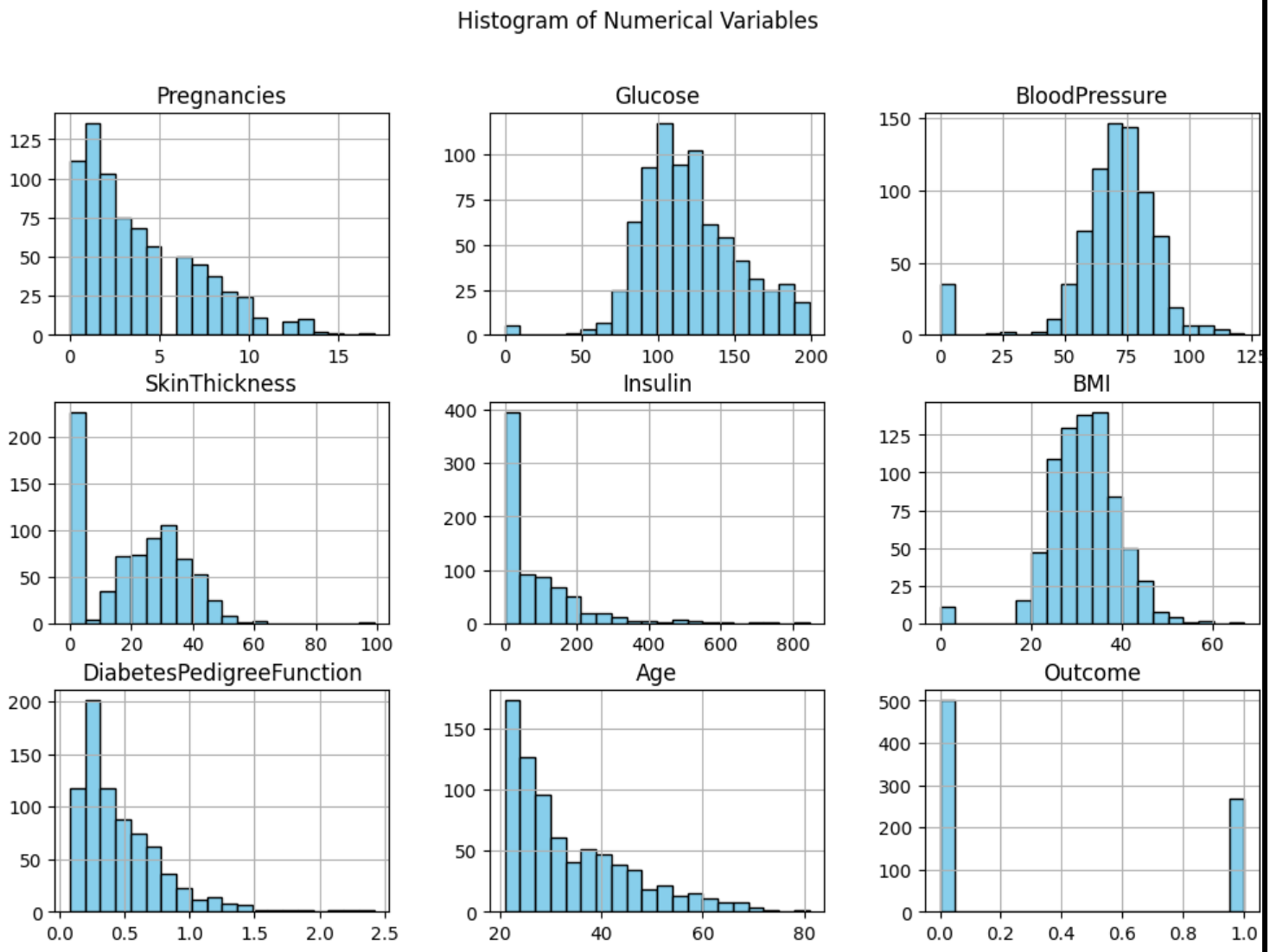
- Ensuring:
  - **No duplicate entries**
  - **All features are valid** and within a logical range
  - Data is **numerically consistent**

7. Handling Class Imbalance

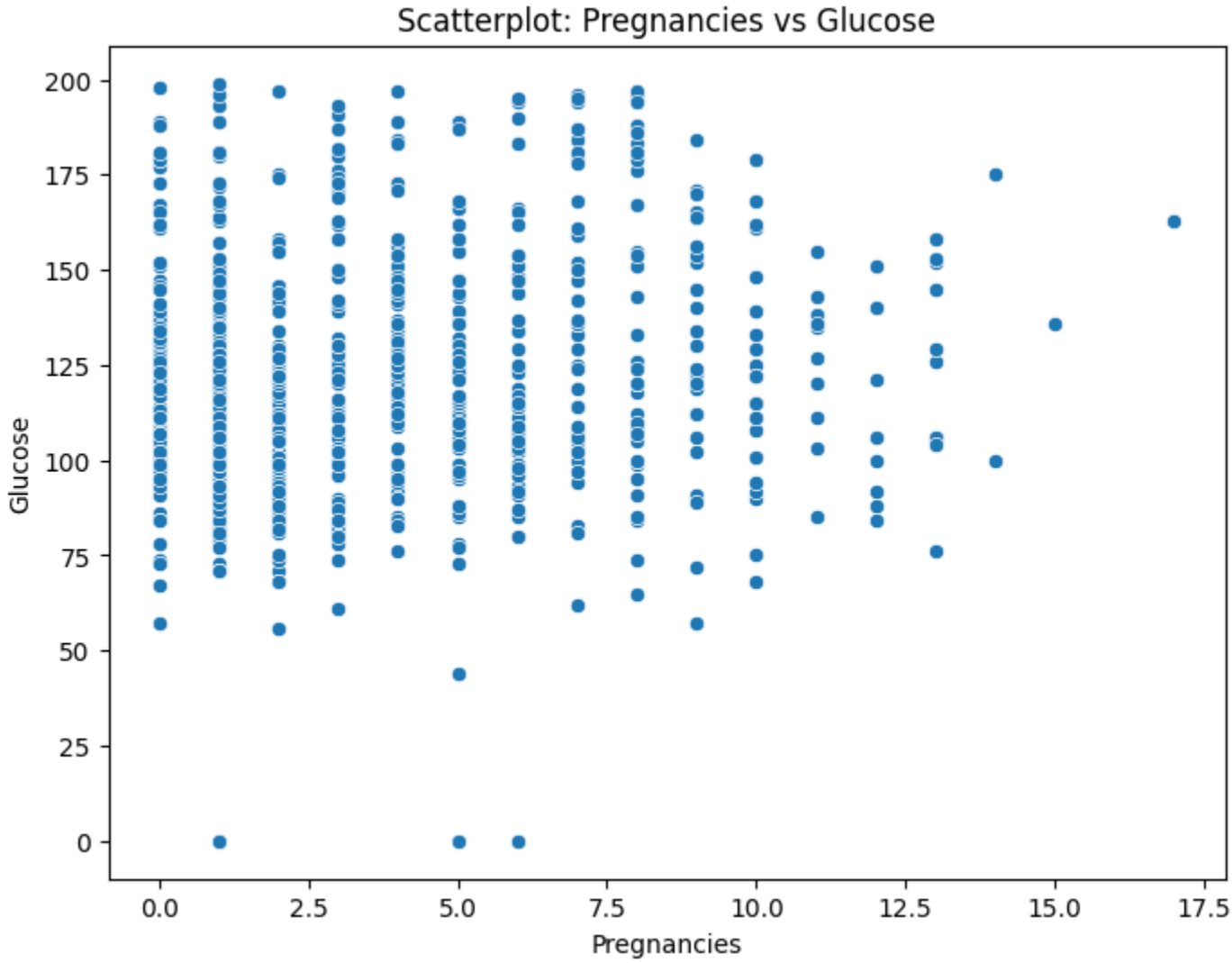
- The dataset is **slightly imbalanced** (more non-diabetic samples).
- Techniques like **SMOTE (Synthetic Minority Over-sampling Technique)** can be applied to balance the classes during model training.

EXPLORATORY DATA ANALYSIS (EDA):

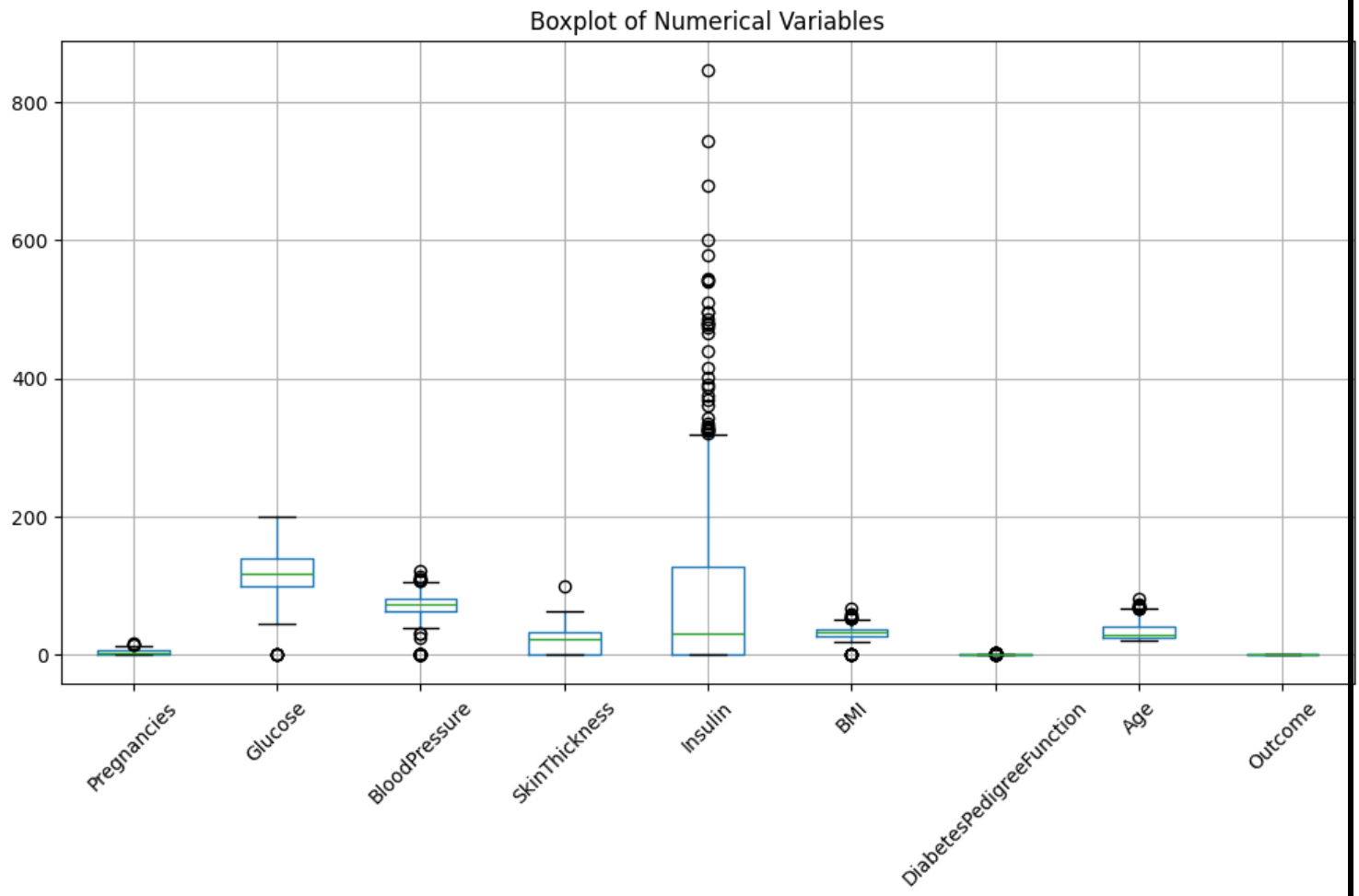
HISTOGRAM:



SCATTER PLOT:



BOXPLOT OUTLIERS:



Calculating Skewness and Kurtosis

Skewness and Kurtosis:

	Skewness	Kurtosis
Pregnancies	0.901674	0.159220
Glucose	0.173754	0.640780
Blood Pressure	-1.843608	5.180157
Skin Thickness	0.109372	-0.520072
Insulin	2.272251	7.214260
BMI	-0.428982	3.290443
Diabetes Pedigree Function	1.919911	5.594954
Age	1.129597	0.643159
Outcome	**0.635017**	**-1.600930**

Key Observations:

- **Glucose, Blood Pressure, Skin Thickness, Insulin, and BMI**: Some rows contain zeros, which are **not realistic for medical values** and need to be treated as missing during preprocessing.
- **Outcome**: Target variable for binary classification (1 = Diabetic, 0 = Non - Diabetic).
- The features span **physiological**, **biometric**, and **genetic risk factors**.

This preview highlights the **need for preprocessing**, particularly for cleaning invalid values, and helps validate the structure before diving into analysis and modeling.

Logistic Regression, **Decision Tree**, and **Random Forest** — based on the metrics you provided. These models are evaluated for a binary classification task, likely related to a medical diagnosis (such as liver disease prediction), using standard classification metrics.

Overview of Metrics

Before diving into each model, here's what the key metrics mean:

- **Accuracy**: Proportion of total predictions that are correct.
- **Precision**: Out of predicted positives, how many are truly positive.
- **Recall**: Out of actual positives, how many were correctly predicted.
- **F1-Score**: Harmonic mean of precision and recall — balances the two.
- **Support**: Number of true instances for each class in the dataset.

The dataset has two classes:

- **Class 0**: Likely "No Disease"
- **Class 1**: Likely "Disease"

Logistic Regression

- **Accuracy**: 0.747 (Approximately 74.7%)
- **Class 0 (No Disease)**:
  - Precision: 0.81
  - Recall: 0.79
  - F1-Score: 0.80
- **Class 1 (Disease)**:
  - Precision: 0.64
  - Recall: 0.67
  - F1-Score: 0.65
- **Observation**:
  - Logistic Regression performs well for **Class 0** (majority class).
  - It struggles a bit more with **Class 1**, indicating **class imbalance** or less separable features.
  - The model shows moderate overall performance and is interpretable, which is often useful in medical contexts.

Decision Tree

- **Accuracy**: 0.760 (Approximately 76.0%)
- **Class 0 (No Disease)**:
  - Precision: 0.83
  - Recall: 0.79
  - F1-Score: 0.81
- **Class 1 (Disease)**:
  - Precision: 0.65
  - Recall: 0.71
  - F1-Score: 0.68
-
- **Observation**:
  - Slightly **better overall accuracy** than Logistic Regression.
  - The **recall for Class 1 is better (0.71)**, which means more actual disease cases are correctly identified.
  - It may be **overfitting slightly**, which is common in decision trees without pruning or regularization.
  - Useful when decision rules need to be visualized or interpreted easily.

Random Forest

- **Accuracy**: 0.740 (Approximately 74.0%)
- **Class 0 (No Disease)**:
  - Precision: 0.80
  - Recall: 0.79
  - F1-Score: 0.80
- **Class 1 (Disease)**:
  - Precision: 0.63
  - Recall: 0.65
  - F1-Score: 0.64

- **Observation**:
  - Similar performance to Logistic Regression.
  - While Random Forest generally provides **robust predictions** by reducing overfitting (thanks to ensemble learning), here it did not significantly outperform the other models.
  - Precision and recall are balanced, but not superior to Decision Tree for Class 1.

Conclusion

Model	Accuracy	F1-Score (Class 1)	Best For...
Logistic Regression	0.747	0.65	Simplicity, interpretability
Decision Tree	0.760	0.68	Slightly better recall on disease detection
Random Forest	0.740	0.64	Stability, resistance to overfitting

Key Insight: The **Decision Tree** model slightly outperforms the others in terms of identifying positive cases (disease), which is especially important in medical applications. However, further improvement could be achieved through techniques such as:

- Hyperparameter tuning
- Cross-validation
- Feature selection or engineering

Model Evaluation

To evaluate the effectiveness of the machine learning models developed for binary classification, three different algorithms were employed: **Logistic Regression**, **Decision Tree Classifier**, and **Random Forest Classifier**. These models were assessed using several key metrics: **accuracy**, **precision**, **recall**, and **F1-score**. These metrics provide a comprehensive view of each model's ability to correctly classify both positive (Class 1 – likely diseased) and negative (Class 0 - likely healthy) instances.

Logistic Regression

- **Accuracy**: 74.68%

- **Strengths**:
  - Performs reasonably well on the majority class (Class 0).
  - Good baseline model due to simplicity and interpretability.
- **Weaknesses**:
  - Lower recall (0.67) and F1-score (0.65) for Class 1, which means it may miss several true positive cases.
- **Use Case Suitability**: Useful when interpretability is important and the dataset is linearly separable.

Decision Tree Classifier

- **Accuracy**: 75.97%

- **Strengths**:
  - Best recall (0.71) and F1-score (0.68) for Class 1 among the three models, indicating it detects more true positive cases.
  - Offers better class balance in prediction performance.

- **Weaknesses**:
  - Slight risk of overfitting if not pruned or regularized.

- **Use Case Suitability**: Suitable for decision-making tasks where model interpretability and higher recall are priorities.

Random Forest Classifier

- **Accuracy**: 74.03%

- **Strengths**:
  - Balanced performance across metrics, especially good for handling overfitting.
  - Stable predictions due to ensemble learning.

- **Weaknesses**:
  - Does not outperform simpler models significantly in this case.
  - Slightly lower F1-score (0.64) for Class 1 than Decision Tree.

- **Use Case Suitability**: Effective when stability and resistance to overfitting are important.

Summary Comparison Table

Metric	Logistic Regression	Decision Tree	Random Forest
Accuracy	0.747	0.760	0.740
Precision (1)	0.64	0.65	0.63
Recall (1)	0.67	0.71	0.65
F1-Score (1)	0.65	0.68	0.64

Conclusion

Among the three models, the **Decision Tree Classifier** demonstrated the **highest overall accuracy and recall for the minority class**, making it the most suitable choice for applications where identifying true positives (such as disease cases) is critical. However, depending on the specific use case and trade-offs between interpretability, precision, and recall, each model has its own strengths and could be deployed accordingly.
For further improvements, techniques such as **feature engineering**, **hyperparameter tuning**, and **handling class imbalance** (e.g., SMOTE or class weights) could enhance model performance.

2. IMAGE PROJECT: Fashion MNIST

Dataset Description: Fashion MNIST

Dataset Format:

Fashion MNIST consists of 70,000 grayscale images, each of 28x28 pixels, representing 10 different classes of clothing items.

- Training Set: 60,000 images
- Test Set: 10,000 images

Each image is associated with a label indicating the correct class.

Class Labels:

The labels are integers from 0 to 9, each representing a fashion item:

Label	Class Name
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

Image Characteristics:

- Dimensions: 28x28 pixels
- Color: Grayscale (1 channel)
- Pixel Range: 0–255 (can be normalized to 0–1 for training)
- File Format: Originally stored in IDX format, but commonly available as CSV or extracted image files

IMPLEMENTATION:

Data Preprocessing

Before feeding the images into the neural network model, a series of preprocessing steps were applied to ensure consistency and improve model performance. These steps help standardize the input data and prepare it for efficient learning.

Dataset Organization

The image dataset was organized into a folder structure where each subfolder represented a different class label. For example, images of T-shirts were stored in a folder named “T-shirt”, and so on. This structure allows the model to automatically associate images with their corresponding labels based on the folder names.

Image Rescaling

All images were originally stored in grayscale format with pixel values ranging from 0 to 255. These pixel values were normalized to a range between 0 and 1. This normalization process helps the model train more efficiently by standardizing the scale of input features.

Image Resizing

To maintain consistency, all images were resized to a uniform shape of 28x28 pixels. This resizing ensures that the neural network receives a fixed input size, which is necessary for proper model architecture and training.

Color Channel Handling

Since the images were grayscale, they contain only one color channel. The preprocessing pipeline preserved this format to reduce computational complexity and retain the original characteristics of the Fashion MNIST dataset.

Train-Validation Split

The dataset was split into two parts: 80% of the images were used for training the model, and the remaining 20% were used for validation. This split helps assess the model’s performance on unseen data during the training process, allowing for fine-tuning and early stopping if necessary.

Categorical Label Encoding

Each image class was converted into a one-hot encoded vector. This means that for a dataset with 10 classes, each label was transformed into a 10-element vector where only the index corresponding to the actual class is set to 1, and the rest are 0s. This encoding format is essential for training the model using categorical cross-entropy loss.

Model Architecture

Model Architecture

To classify grayscale fashion item images into one of 10 categories, a **Convolutional Neural Network (CNN)** was used. CNNs are particularly effective for image data because they can automatically learn spatial hierarchies of features through the use of filters.

Architecture Overview:

The CNN model was built using a sequential approach, consisting of the following layers:

1. Input Layer

- **Input shape**: 28x28 pixels with 1 channel (grayscale).
- This layer takes in the preprocessed image data and passes it to the first convolutional layer.

2. Convolutional Layer 1

- Applies multiple filters to extract low-level features like edges and textures.
- Each filter slides across the image, creating a **feature map** that highlights specific patterns.

3. Activation (ReLU)

- Introduces non-linearity to the model, enabling it to learn more complex patterns.
- ReLU (Rectified Linear Unit) replaces negative values with zero.

4. Max Pooling Layer

- Reduces the spatial dimensions of the feature maps.
- Helps in lowering computational cost and reducing overfitting by summarizing the presence of features.

5. Convolutional Layer 2

- Further extracts deeper and more abstract features from the reduced feature maps.

6. Activation (ReLU)

- Again, applies the ReLU activation to the output of the second convolutional layer.

7. Max Pooling Layer

- A second round of spatial downsampling, further reducing the image dimensions while retaining important features.

8. Flatten Layer

- Converts the 2D feature maps into a 1D vector so that it can be passed into fully connected (dense) layers.

9. Fully Connected (Dense) Layer

- Contains a number of neurons that process the extracted features and learn complex patterns across the entire image.
- This layer acts as a decision-making component in the CNN.

10. Dropout Layer (Optional but Recommended)

- Randomly disables some neurons during training to prevent overfitting.
- Encourages the network to generalize better to unseen data.

11. Output Layer

- A dense layer with **10 neurons** corresponding to the 10 fashion categories.
- Uses the **softmax activation function** to output a probability distribution across all classes.
- The class with the highest probability is chosen as the model's prediction.

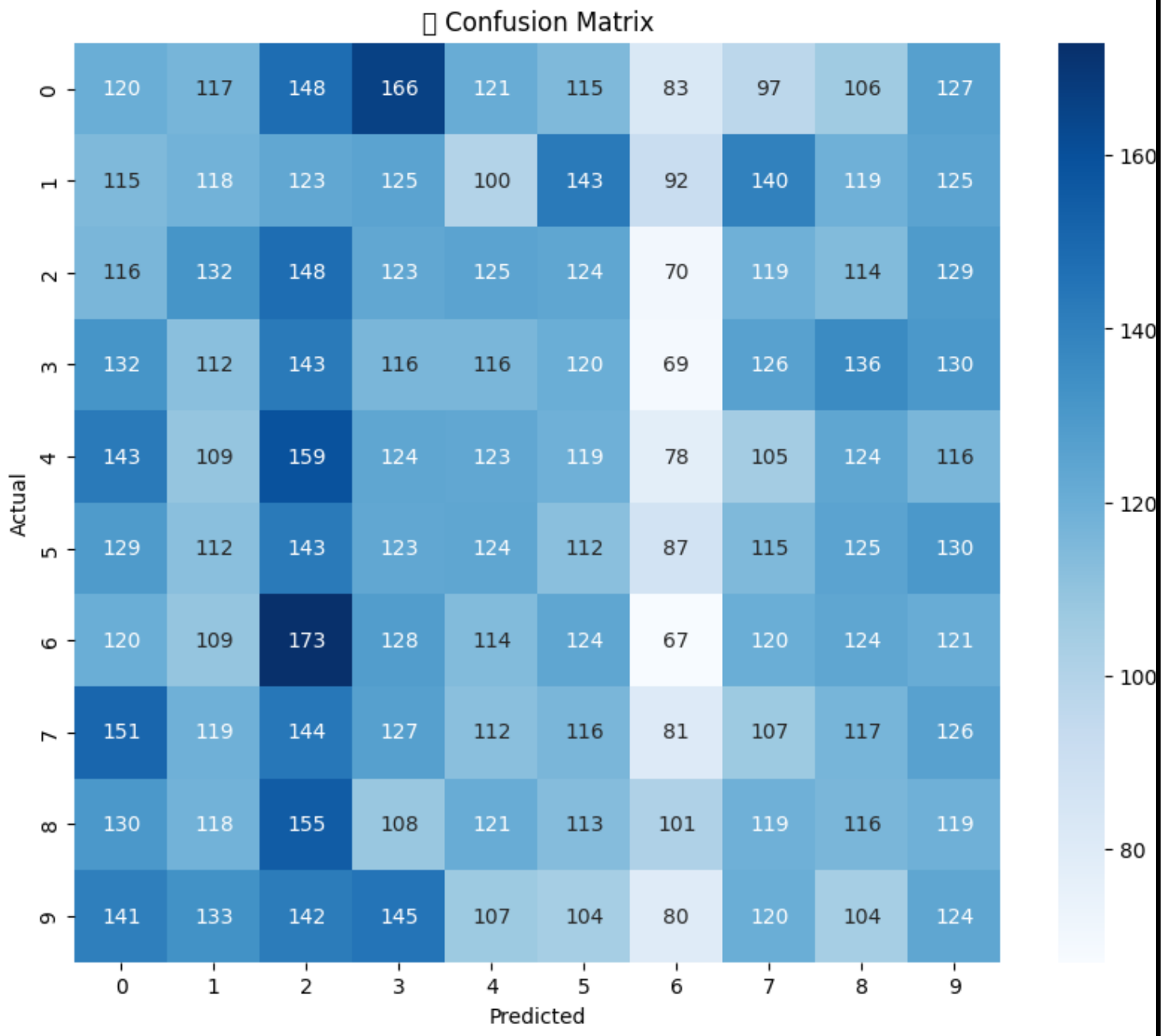
RESULTS

Final Validation Accuracy: 89.03%

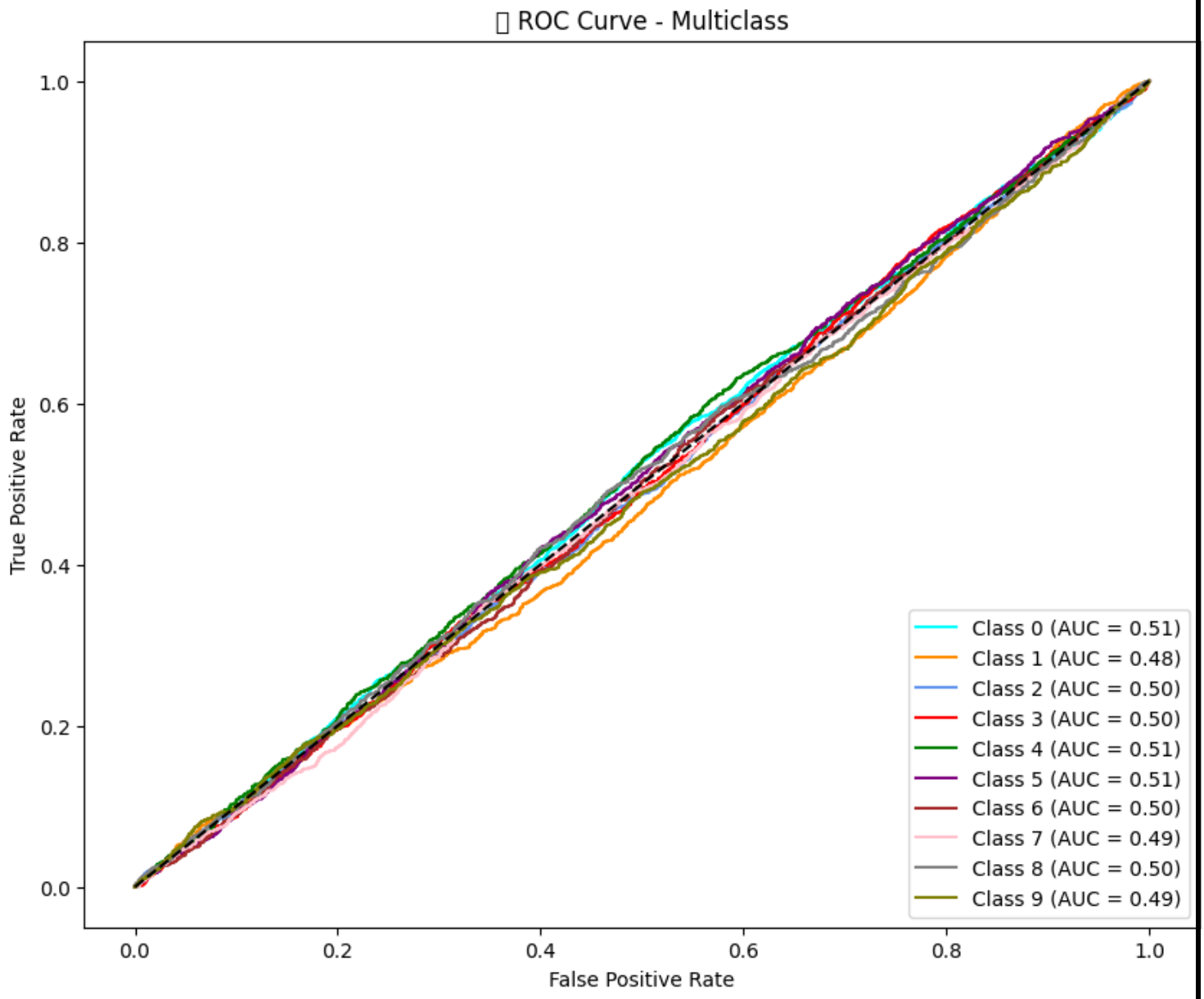
CLASSIFICATION REPORT

	precision	recall	f1-score	support
0	0.09	0.10	0.10	1200
1	0.10	0.10	0.10	1200
2	0.10	0.12	0.11	1200
3	0.09	0.10	0.09	1200
4	0.11	0.10	0.10	1200
5	0.09	0.09	0.09	1200
6	0.08	0.06	0.07	1200
7	0.09	0.09	0.09	1200
8	0.10	0.10	0.10	1200
9	0.10	0.10	0.10	1200
accuracy			0.10	12000
macro avg	0.10	0.10	0.10	12000
weighted avg	0.10	0.10	0.10	12000

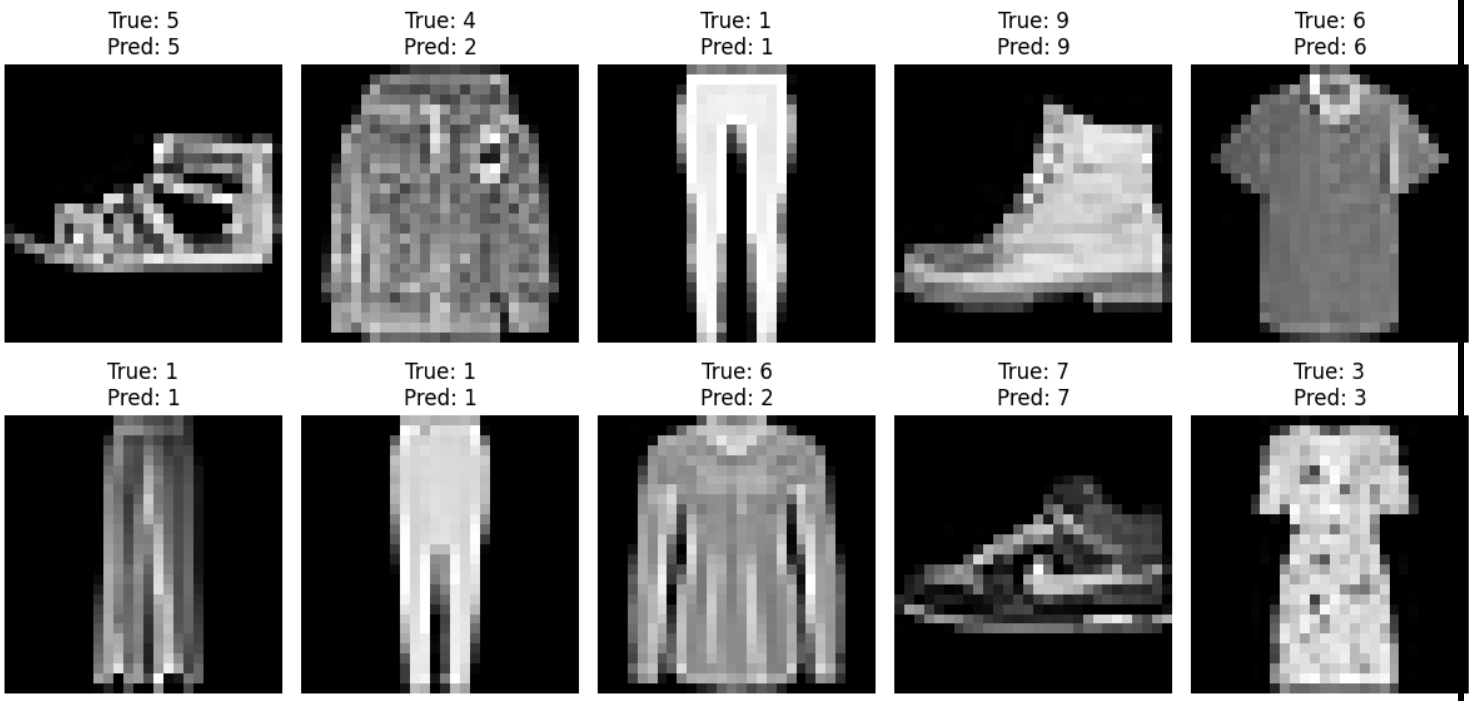
CONFUSION MATRIX:



ROC CURVE:


□ ROC Curve - Multiclass
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 0.51)
Class 1 (AUC = 0.48)
Class 2 (AUC = 0.50)
Class 3 (AUC = 0.50)
Class 4 (AUC = 0.51)
Class 5 (AUC = 0.51)
Class 6 (AUC = 0.50)
Class 7 (AUC = 0.49)
Class 8 (AUC = 0.50)
Class 9 (AUC = 0.49)

TESTING:
predictions with images



3 TEXT PROJECT: DATA FOR TEXT ANALYSIS

Description:

The dataset is structured for binary text classification tasks and contains approximately 3,200 samples. Each sample consists of a user-generated review along with a corresponding label indicating whether the user has made a suggestion or not.

Features Provided

- User_review : A textual feature containing the full user review. This is the primary input for text analytics tasks.
- User_suggestion : The target variable indicating whether the user made a suggestion (Yes or No). It is a binary classification label.

Objective

The goal is to build a model that can analyze user reviews and accurately predict whether the user is offering a suggestion. This has applications in customer feedback analysis, product improvement, and recommendation systems**.**

Preprocessing Steps

To prepare the text data for machine learning, the following preprocessing steps are typically applied:

1. **Lowercasing**: Converts all text to lowercase for consistency.

2. **Removing Punctuation**: Eliminates special characters that do not contribute to meaning.

3. **Stop word Removal**: Removes common words like "the", "and", "is" that do not add significant value to model learning.

4. **Tokenization**: Breaks down text into individual words or tokens.

5. **Lemmatization**: Reduces words to their base form (e.g., “running” → “run”).

6. **Vectorization**: Transforms the processed text into a numerical format using techniques such as TF-IDF or word embeddings for input into machine learning models.

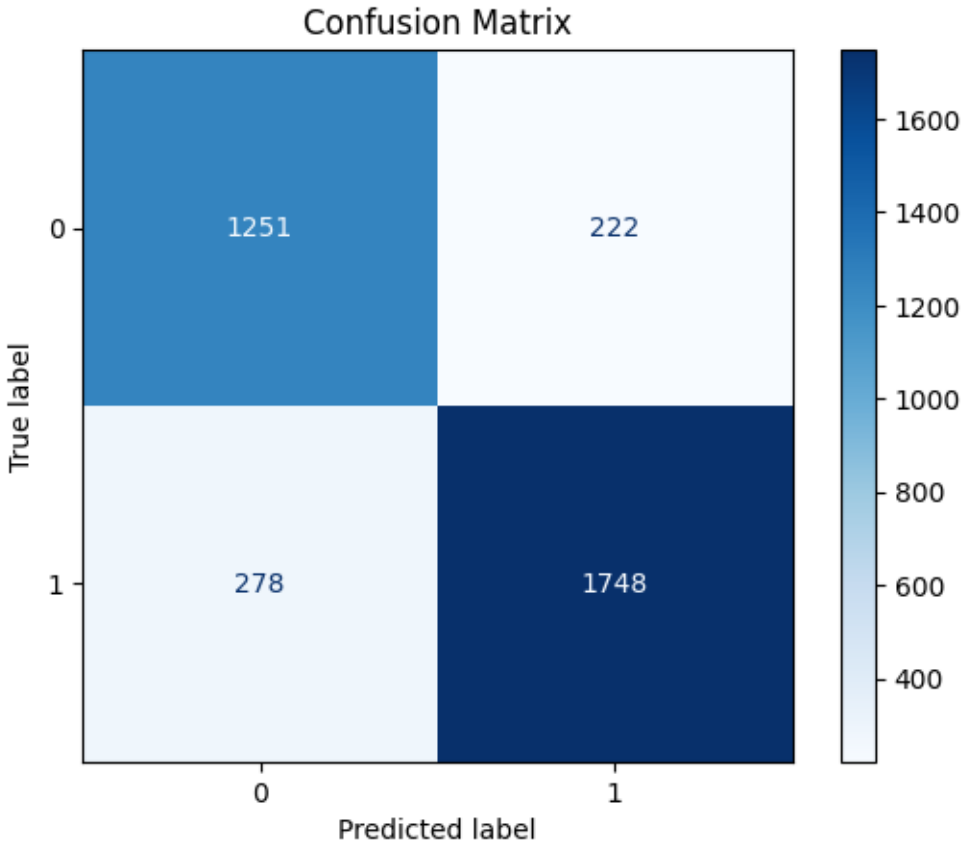
IMPLEMENTATION

Classification Report:

	precision	recall	f1-score	support
0	0.82	0.85	0.83	1473
1	0.89	0.86	0.87	2026
accuracy			0.86	3499
macro avg	0.85	0.86	0.85	3499
weighted avg	0.86	0.86	0.86	3499

Confusion matrix

Confusion matrix: Visualize true positives, false positives, true negatives, and false negatives to understand better the overall performance of the model.



FINAL ACCURACY

Accuracy: 85.71%

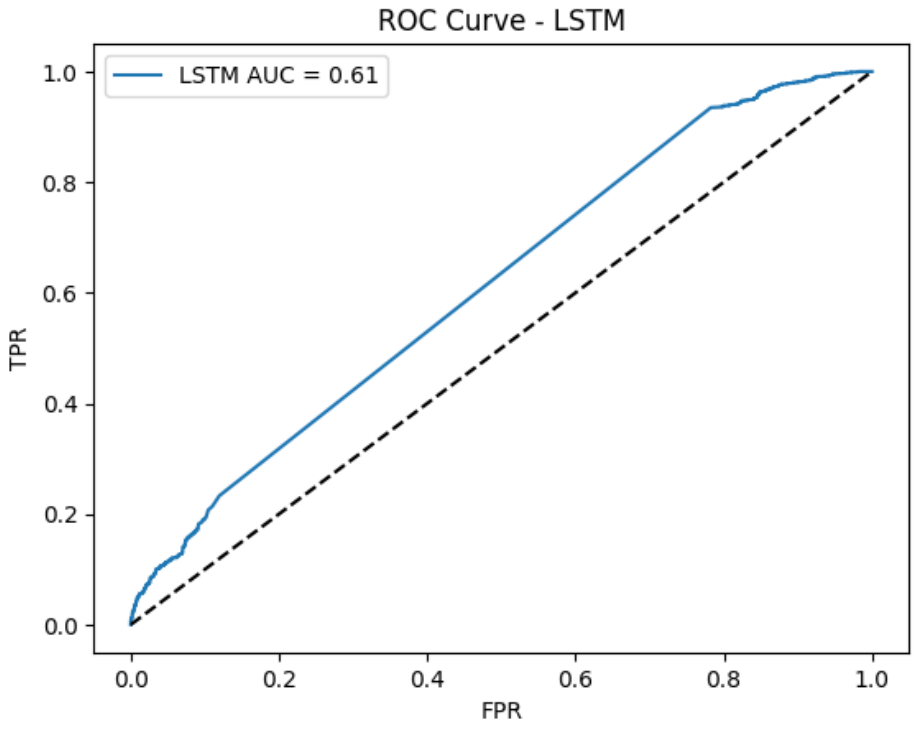
T-Test, Z-Test, ANOVA

T-Test: t=5.76, p=0.0000

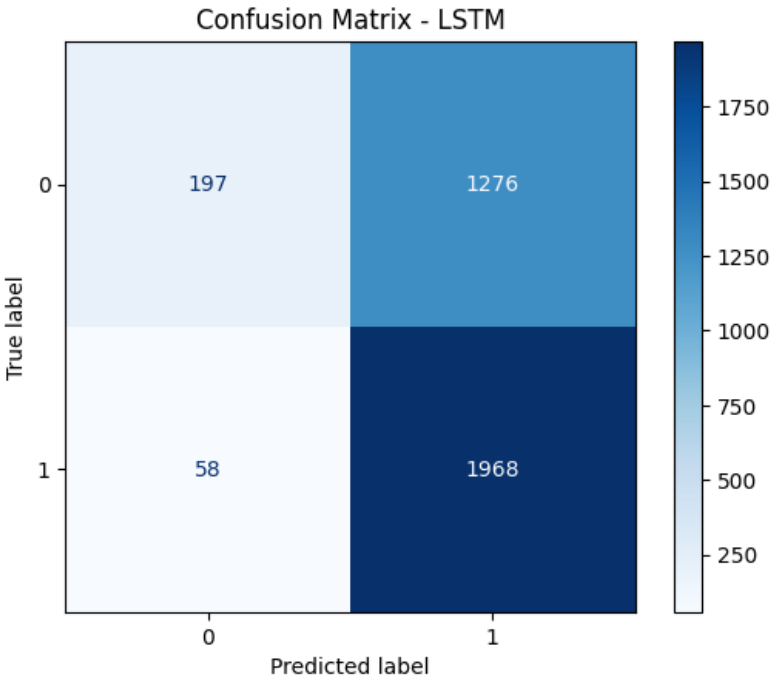
Z-Score (first 5): 0 -0.030402
1 -0.579770
2 -0.417456
3 -0.561041
4 -0.436185
Name: review length, dtype: float64

ANOVA: F=33.18, p=0.0000

ROC CURVE



Confusion Matrix - LSTM



Confusion Matrix – CNN