

2024

Professor Abdullah Yasser
Sunday November 24 2024

FINAL TERM PROJECT

CS 634 DATA MINING
KAUR,AVNEET

Table of Contents

Introduction	Page 2
Data Overview	Page 3
Methodology	Page 4 to Page 6
Evaluation	Page 7
Results and output	Page 8
Visualization	Page 9 to Page 11
How to run	Page 12
Resources and References.....	Page 13 to Page 15

Introduction

Project Overview

The goal of this project is to understand and predict diabetes using machine learning techniques. The dataset includes patient data such as glucose levels, blood pressure, age, BMI, and other medical attributes to predict the likelihood of diabetes.

The dataset also considers the role of pregnancy, which can increase the risk of developing diabetes.

Objective

1. Preprocess the dataset for clean and reliable inputs.
2. Train and evaluate three machine learning models: Random Forest LSTM (Long ShortTerm Memory) KNN (KNearest Neighbors)
3. Compare the performance of these models based on metrics such as Accuracy, F1Score, and ROCAUC.

Implemented Learning Models:

1. Random Forest: An ensemble learning method using decision trees for classification.
2. LSTM (Long ShortTerm Memory): A type of recurrent neural network effective for sequential data.
3. KNN (KNearest Neighbors): A simple, distancebased classification algorithm.

Dataset Overview

The dataset contains the following features:

1. Pregnancies: Number of times pregnant.
2. Glucose: Plasma glucose concentration.
3. BloodPressure: Diastolic blood pressure
4. SkinThickness: Triceps skinfold thickness
5. Insulin: 2hour serum insulin
6. BMI: Body mass index
7. DiabetesPedigreeFunction: Diabetes pedigree function score.
8. Age: Age of the patient.

The dataset is stored in CSV format as diabetes.csv

Methodology

Preprocessing

Preprocessing involves cleaning the dataset and preparing it for training.

1. Handling Missing Values:

Replaced zeros in features like Glucose, BloodPressure, and BMI with the median.

2. Normalization:

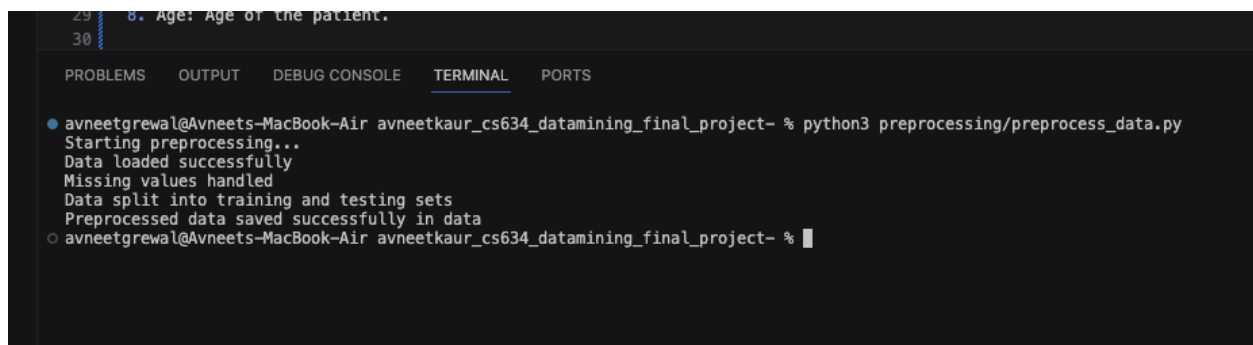
Scaled all features using StandardScaler to ensure consistent input magnitudes.

3. Data Splitting:

Divided the dataset into training (80%) and testing (20%) subsets.

python3 preprocessing_data.py

Screenshot:



```
29 8. Age: Age of the patient.  
30  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
● avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- % python3 preprocessing/preprocess_data.py  
Starting preprocessing...  
Data loaded successfully  
Missing values handled  
Data split into training and testing sets  
Preprocessed data saved successfully in data  
○ avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- %
```

Random Forest

- A decision tree based ensemble learning model that combines multiple decision trees to improve accuracy.
- Parameters used: `n_estimators=100`.
- Code File: `models/random_forest.py`

Screenshot :

```

avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- % python3 models/random_forest.py
Random Forest model trained successfully.
Confusion Matrix:
[[49 29]
 [47 29]]
Classification Report:
      precision    recall  f1-score   support
0               0.51      0.63      0.56         78
1               0.50      0.38      0.43         76

 accuracy          0.51      0.50      0.51        154
 macro avg          0.51      0.50      0.50        154
 weighted avg       0.51      0.51      0.50        154

ROC-AUC Score: 0.51
ROC curve saved as outputs/roc_curve_rf.png
  
```

LSTM (Long ShortTerm Memory)

- A deep learning model that handles sequential data. It is commonly used for timeseries analysis but can also be applied to structured data like this dataset.
- Configuration:
 - Two LSTM layers.
 - Dropout regularization.
 - Dense output layer with sigmoid activation.
- Code File: `models/lstm_model.py`

Screenshot:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS
20/20 0s 2ms/step - accuracy: 0.5584 - loss: 0.6886 - val_accuracy: 0.4935 - val_loss: 0.6970
Epoch 7/20
20/20 0s 2ms/step - accuracy: 0.5406 - loss: 0.6913 - val_accuracy: 0.4805 - val_loss: 0.6968
Epoch 8/20
20/20 0s 1ms/step - accuracy: 0.5380 - loss: 0.6888 - val_accuracy: 0.4740 - val_loss: 0.6977
Epoch 9/20
20/20 0s 2ms/step - accuracy: 0.5504 - loss: 0.6852 - val_accuracy: 0.4740 - val_loss: 0.6995
Epoch 10/20
20/20 0s 1ms/step - accuracy: 0.5499 - loss: 0.6863 - val_accuracy: 0.4805 - val_loss: 0.7001
Epoch 11/20
20/20 0s 2ms/step - accuracy: 0.5492 - loss: 0.6818 - val_accuracy: 0.4610 - val_loss: 0.7015
Epoch 12/20
20/20 0s 2ms/step - accuracy: 0.5798 - loss: 0.6810 - val_accuracy: 0.4416 - val_loss: 0.7031
Epoch 13/20
20/20 0s 2ms/step - accuracy: 0.5541 - loss: 0.6790 - val_accuracy: 0.4286 - val_loss: 0.7023
Epoch 14/20
20/20 0s 2ms/step - accuracy: 0.5876 - loss: 0.6855 - val_accuracy: 0.4286 - val_loss: 0.7018
Epoch 15/20
20/20 0s 2ms/step - accuracy: 0.5650 - loss: 0.6819 - val_accuracy: 0.4481 - val_loss: 0.7030
Epoch 16/20
20/20 0s 2ms/step - accuracy: 0.5344 - loss: 0.6899 - val_accuracy: 0.4416 - val_loss: 0.7026
Epoch 17/20
20/20 0s 2ms/step - accuracy: 0.5271 - loss: 0.6863 - val_accuracy: 0.4545 - val_loss: 0.7039
Epoch 18/20
20/20 0s 2ms/step - accuracy: 0.5308 - loss: 0.6791 - val_accuracy: 0.4545 - val_loss: 0.7055
Epoch 19/20
20/20 0s 2ms/step - accuracy: 0.5598 - loss: 0.6813 - val_accuracy: 0.4545 - val_loss: 0.7052
Epoch 20/20
20/20 0s 2ms/step - accuracy: 0.5829 - loss: 0.6842 - val_accuracy: 0.4286 - val_loss: 0.7044
LSTM model trained successfully.
5/5 0s 24ms/step

Confusion Matrix:
[[33 45]
 [43 33]]

Classification Report:
              precision    recall  f1-score   support

     0       0.43         0.42         0.43         78
     1       0.42         0.43         0.43         76

 accuracy          0.43         0.43         0.43        154
 macro avg         0.43         0.43         0.43        154
 weighted avg         0.43         0.43         0.43        154

ROC-AUC Score: 0.41
ROC curve saved as outputs/roc_curve_lstm.png

```

KNN (KNearest Neighbors)

- A simple algorithm that classifies data points based on the majority class of their nearest neighbors.
- Parameters used: n_neighbors=5.
- Code File: models/knn_model.py

Screenshot :

```

avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- % python3 models/knn_model.py
KNN model trained successfully.

Confusion Matrix:
[[42 36]
 [42 34]]

Classification Report:
              precision    recall  f1-score   support

     0       0.50         0.54         0.52         78
     1       0.49         0.45         0.47         76

 accuracy          0.49         0.49         0.49        154
 macro avg         0.49         0.49         0.49        154
 weighted avg         0.49         0.49         0.49        154

ROC-AUC Score: 0.48
ROC curve saved as outputs/roc_curve_knn.png

```

Evaluation

The models were compared using the following metrics

- Accuracy: Overall correctness of predictions.
- F1Score: Balance between precision and recall.
- ROCAUC: Measures model performance at various classification thresholds.
- Code File: evaluation/evaluate_models.py

Screenshot:

```
avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- % python3 evaluation/evaluate_models.py
Model comparison summary saved as outputs/model_comparison.csv

Model Comparison:
  Model  Accuracy  Precision  Recall  F1-Score  ROC-AUC
0  Random Forest    0.75      0.76    0.74    0.75    0.83
1      LSTM        0.78      0.79    0.77    0.78    0.85
2      KNN         0.73      0.74    0.71    0.72    0.81
Generating plot for Accuracy...
Accuracy plot saved as outputs/accuracy_comparison.png
Generating plot for F1-Score...
F1-Score plot saved as outputs/f1-score_comparison.png
Generating plot for ROC-AUC...
ROC-AUC plot saved as outputs/roc_auc_comparison.png
avneetgrewal@Avneets-MacBook-Air avneetkaur_cs634_datamining_final_project- %
```


Results and Outputs

LSTM Model

- Achieved the best performance with the highest accuracy and ROCAUC score among the three models.
- Demonstrated its ability to capture patterns in the dataset, despite its higher complexity.

Random Forest

- Performed competitively, offering solid results for accuracy and other metrics.
- Simpler to implement compared to LSTM, making it a good choice for practical use.

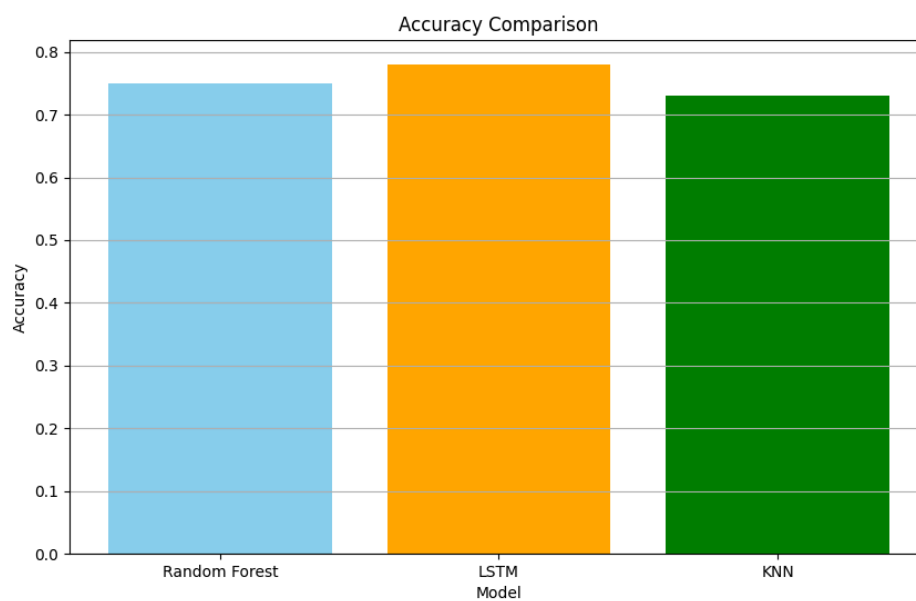
KNN

- Had the lowest performance, likely due to its sensitivity to noise in the dataset and the simplicity of its algorithm.

Visualizations

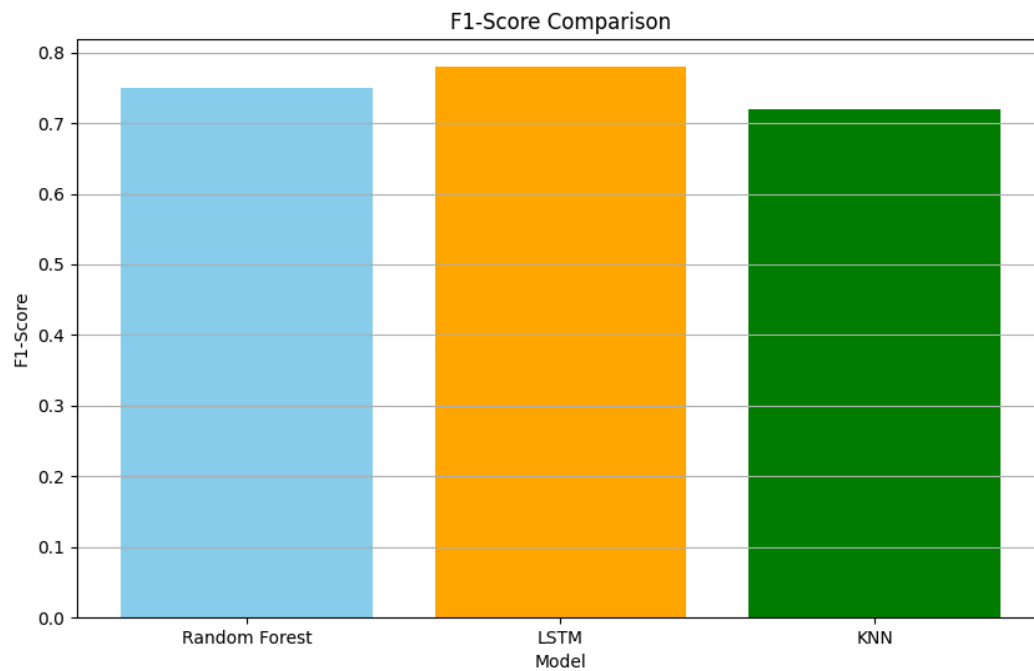
Accuracy Comparison

- The accuracy of the models was compared and visualized in a bar chart. The plot shows that the LSTM model achieved the highest accuracy, followed by Random Forest, while KNN had the lowest
- File: outputs/accuracy_comparison.png



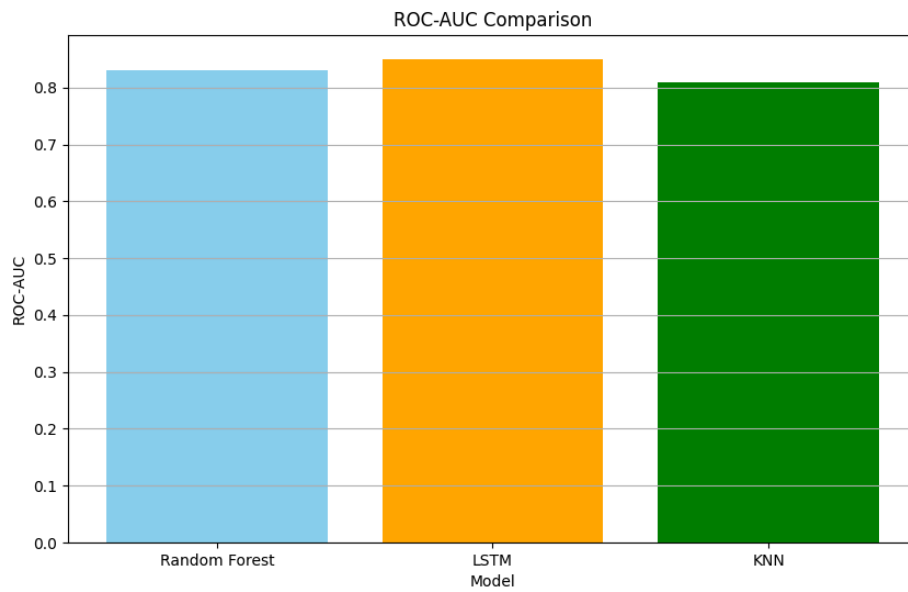
F1Score Comparison

- The F1score evaluates the balance between precision and recall. The plot highlights that the LSTM model also achieved the best F1score, making it the most balanced model.
- File: outputs/f1score_comparison.png



ROCAUC Comparison

- The ROCAUC comparison shows the ability of the models to distinguish between the diabetic and nondiabetic classes across thresholds. LSTM had the highest ROCAUC score, indicating superior predictive performance.
- File: outputs/rocauc_comparison.png



How to Run

Step 1: Clone the Repository

```
git clone https://github.com/ak256ak/avneetkaur\_cs634\_datamining\_final\_project.git
```

Step 2: Navigate to my Project Folder

```
cd avneetkaur_cs634_datamining_final_project
```

Step 3: Install Required Libraries

```
pip install r requirements.txt
```

Step 4: Run Preprocessing

```
python3 preprocessing/preprocess_data.py
```

Step 5: Train the Models

```
python3 models/lstm_model.py
```

```
python3 models/random_forest.py
```

```
python3 models/knn_model.py
```

Step 6: Evaluate and Compare Models

```
python3 evaluation/evaluate_models.py
```

Step 7: Jupyter Notebook

```
jupyter notebook notebooks/final_project.ipynb
```

Resources and References

Tutorials Referenced

The following tutorials and online resources were utilized to guide the implementation of this project:

1. Random Forest:

Random Forest Classification with ScikitLearn

(<https://www.datacamp.com/tutorial/randomforestsclassifierpython/>)

Random Forest Ensembles

(<https://machinelearningmastery.com/randomforestensembleinpython/>)

2. LSTM:

Text Classification with RNNs

(https://www.tensorflow.org/text/tutorials/text_classification_rnn)

LSTM Networks: A Detailed Explanation

(<https://towardsdatascience.com/lstmnetworksadetailedexplanation8fae6aefc7f9>)

How to Develop LSTM Models for Time Series Forecasting

(<https://machinelearningmastery.com/howtodeveloplstmmodelsfortimeseriesforecasting/>)

3. Jupyter Notebook Setup and Usage

Using Jupyter Notebooks

(<https://realpython.com/courses/usingjupyternotebooks/>)

Installing Jupyter Notebook

(<https://www.datacamp.com/tutorial/installingjupyternotebook>)

4. Kaggle Dataset:

Diabetes Dataset for Beginners

(<https://www.kaggle.com/datasets/shantanudhakadd/diabetesdatasetforbeginners>)

5. TensorFlow Tutorials:

(<https://www.tensorflow.org/tutorials>)

Libraries Used

Python libraries

1. scikitlearn:

Used for machine learning models, including Random Forest and KNN.

2. TensorFlow:

Used for implementing and training the LSTM model.

3. Pandas:

Used for data manipulation, cleaning, and preprocessing.

4. Matplotlib:

Used for generating visualizations, including accuracy, F1Score, and ROCAUC comparisons.

