DATA MINING
FINALTERM PROJECT

NAME: CHAITANYA NANDINI MINNEKANTI

NJIT ID: 31651119

NJIT UCID: cm723

EMAIL ADDRESS: cm723@njit.edu

Supervised Data Mining Project Documentation

- Project Title:

"Supervised Data Mining (Classification) - Binary Classification Only"

- Goals and Objectives

Motivation:

The motivation behind this project is to implement and compare three classification algorithms—Random Forest, Support Vector Machine (SVM), and Long Short-Term Memory (LSTM)—to predict diabetes based on medical records. The project aims to explore the strengths and weaknesses of each algorithm in binary classification tasks.

Significance:

Understanding the performance of different classification algorithms is crucial in medical prediction scenarios. Predicting diabetes accurately can aid in early intervention and personalized patient care.

Objectives:

- Implement Random Forest, SVM, and LSTM algorithms for binary classification.
- Evaluate classification performance using manual calculation of metrics.
- Utilize the Pima Indians Diabetes Database dataset for experimentation.
- Employ 10-fold cross-validation for robust model evaluation.
- Present experimental results and compare the algorithms.

Features:

Predictive Variables: Number of pregnancies, BMI, insulin level, age, diabetes pedigree function, skin thickness, blood pressure, pregnancies, glucose.

Target Variable: Outcome (0 or 1)

- Dataset:

Name: Pima Indians Diabetes Database

Source: Kaggle

Description: The dataset comprises medical predictor variables and an outcome variable, indicating whether a patient has diabetes or not.

- ## Detail Design of Features

Analysis

Exploratory Data Analysis (EDA):

- Analyze the distribution of individual variables.
- Identify correlations between features.
- Missing Values Handling:

Handle missing values appropriately for each feature.

- ## Implementation:

Data collection and preprocessing.
Exploratory Data Analysis (EDA).
Algorithm implementation.
Evaluation metrics calculation.
Results presentation.

- ## Algorithm Implementation:

Random Forest

```
#Implementing Random Forest Classifier
rf_classifier = RandomForestClassifier(random_state=42)
rf_classifier.fit(X_train, y_train)
rf_predictions = rf_classifier.predict(X_test)
```

Python

Results:

Tabulate and present the 10-fold cross-validation results.

Results for each run:

	Classifier	Run	TP	TN	FP	FN	Accuracy	Precision	Recall	F1 Score
0	Random Forest	1	16	40	10	11	0.727273	0.615385	0.592593	0.603774
1	Random Forest	2	13	46	4	14	0.766234	0.764706	0.481481	0.590909
2	Random Forest	3	15	40	10	12	0.714286	0.600000	0.555556	0.576923
3	Random Forest	4	19	46	4	8	0.844156	0.826087	0.703704	0.760000
4	Random Forest	5	18	44	6	9	0.805195	0.750000	0.666667	0.705882
5	Random Forest	6	15	44	6	12	0.766234	0.714286	0.555556	0.625000
6	Random Forest	7	14	45	5	13	0.766234	0.736842	0.518519	0.608696
7	Random Forest	8	17	39	11	10	0.727273	0.607143	0.629630	0.618182
8	Random Forest	9	18	40	10	8	0.763158	0.642857	0.692308	0.666667
9	Random Forest	10	12	40	10	14	0.684211	0.545455	0.461538	0.500000

Support Vector Machine (SVM)

```python
# Implementing SVM Classifier
svm_classifier = SVC(random_state=42)
svm_classifier.fit(X_train_scaled, y_train)
svm_predictions = svm_classifier.predict(X_test_scaled)
```

Python

Results:

Tabulate and present the 10-fold cross-validation results.

10	SVM	1	13	47	3	14	0.779221	0.812500	0.481481	0.604651
11	SVM	2	11	46	4	16	0.740260	0.733333	0.407407	0.523810
12	SVM	3	15	44	6	12	0.766234	0.714286	0.555556	0.625000
13	SVM	4	15	49	1	12	0.831169	0.937500	0.555556	0.697674
14	SVM	5	14	47	3	13	0.792208	0.823529	0.518519	0.636364
15	SVM	6	10	45	5	17	0.714286	0.666667	0.370370	0.476190
16	SVM	7	8	47	3	19	0.714286	0.727273	0.296296	0.421053
17	SVM	8	17	43	7	10	0.779221	0.708333	0.629630	0.666667
18	SVM	9	15	40	10	11	0.723684	0.600000	0.576923	0.588235
19	SVM	10	12	45	5	14	0.750000	0.705882	0.461538	0.558140

Long Short-Term Memory (LSTM)

```python
#Building the LSTM model
model = Sequential()
model.add(LSTM(50, input_shape=(X_train.shape[1], X_train.shape[2])))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Python

Results:

Tabulate and present the 10-fold cross-validation results.

```
Results for each run (LSTM):
  Classifier Run  TP  TN  FP  FN  Accuracy  Precision    Recall  F1 Score
0       LSTM   1   9  44   6  18  0.688312   0.600000  0.333333  0.428571
1       LSTM   2   9  44   6  18  0.688312   0.600000  0.333333  0.428571
2       LSTM   3  10  46   4  17  0.727273   0.714286  0.370370  0.487805
3       LSTM   4  12  50   0  15  0.805195   1.000000  0.444444  0.615385
4       LSTM   5  14  46   4  13  0.779221   0.777778  0.518519  0.622222
5       LSTM   6  17  43   7  10  0.779221   0.708333  0.629630  0.666667
6       LSTM   7  19  41   9   8  0.779221   0.678571  0.703704  0.690909
7       LSTM   8  15  44   6  12  0.766234   0.714286  0.555556  0.625000
8       LSTM   9  21  37  13   5  0.763158   0.617647  0.807692  0.700000
9       LSTM  10  21  36  14   5  0.750000   0.600000  0.807692  0.688525
```

Average Results:

```
Average results across all runs:
      Classifier  Accuracy  Precision    Recall  F1 Score
0  Random Forest  0.756425   0.680276  0.585755  0.625603
1            SVM  0.759057   0.742930  0.485328  0.579778
```

```
Average results across all runs (LSTM):
  Classifier  Accuracy  Precision    Recall  F1 Score
0       LSTM  0.752614    0.70109  0.550427  0.595365
```

- Evaluation Metrics

Random Forest Metrics:

{

'TP': 16.1,

'TN': 43.0,

'FP': 7.0,

'FN': 10.7,

'Sensitivity': 0.6008547008547009,

'Specificity': 0.8600000000000001,

'Precision': 0.6988599082077342,

'NPV': 0.8018298200194487,

'TPR': 0.6008547008547009,

'FPR': 0.14000000000000004,

'FNR': 0.39914529914529917,

'TNR': 0.8600000000000001,

'Accuracy': 0.7694976076555025,

'F1 Score': 0.6438294475379698

}

SVM Metrics:

{

'TP': 15.4,

'TN': 43.8,

'FP': 6.2,

'FN': 11.4,

'Sensitivity': 0.5749287749287749,

'Specificity': 0.876,

'Precision': 0.7272573598777392,

'NPV': 0.7951814381600174,

'TPR': 0.5749287749287749,

'FPR': 0.12400000000000003,

'FNR': 0.4250712250712251,

'TNR': 0.876,

'Accuracy': 0.7707792207792208,

'F1 Score': 0.6349028745180703

}

LSTM Metrics:

{'TP': 11.7,

'TN': 38.8,

'FP': 11.2,

'FN': 15.1,

'Sensitivity': 0.43603988603988597,

'Specificity': 0.776,

'Precision': 0.5162974640903334,

'NPV': 0.7225496867721679,

'TPR': 0.43603988603988597,

'FPR': 0.22400000000000003,

'FNR': 0.5639601139601139,

'TNR': 0.776,

'Accuracy': 0.6576213260423788,

'F1 Score': 0.46283399069655334

}

- Discussion:

Accuracy: Random Forest (76.95%) and SVM (77.08%) outperform LSTM (65.76%).

Precision: Random Forest (69.89%) and SVM (72.73%) have higher precision than LSTM (51.63%).

Recall (Sensitivity/True Positive Rate): Random Forest (60.09%) and SVM (57.49%) have higher recall than LSTM (43.60%).

F1 Score: Random Forest (64.38%) and SVM (63.49%) have higher F1 scores than LSTM (46.28%).

Based on these metrics, both Random Forest and SVM perform better than LSTM for this binary classification problem. Random Forest and SVM show better overall accuracy, precision, recall, and F1 score. Random Forest appears to have a balanced performance across different metrics.