Name: Gavin Pinto
NJIT UCID: gkp2
Email Address: gkp2@njit.edu
Date: 24 November 2024
Professor: Yasser Abduallah
CS 634 101 Data Mining

Final Term Project Report

Implementation and Code Usage

Using Random Forest, SVM, and LSTM to Predict Concrete Compressive Strength

Abstract:

In this project, three models were used to make predictions on the Concrete Compressive Strength dataset. The models used were the Random Forest, Support Vector Machine (SVM), and Long Short Term Memory (LSTM). They were then used to predict the Concrete compressive strength based on carefully chosen attributes. Through the use of python libraries and algorithms, the three models were evaluated on the data set using various performance metrics.

Introduction:

Concrete is by far the most important material in civil engineering. Concrete spans countless systems, and the strength of concrete can determine the usability and safety of infrastructures all over the world. The Concrete Compressive Strength dataset demonstrates that the strength of concrete, measured as compressive strength, is a highly nonlinear function of its attributes, which include cement, blast furnace slag, fly ash, water, superplasticizer, coarse aggregate, fine aggregate, and age.

Although all these attributes are correlated with concrete compressive strength, it is redundant and actually detrimental to use all of them in predicting the target variable. A correlation matrix is used to find the attributes with the most correlation with the target variable in order to be selected. Also, one model can be insufficient in providing an optimal relationship between the selected attributes and the concrete compressive strength. Thus, the Random Forest, Support Vector Machine (SVM), and Long Short Term Memory (LSTM) models are used to predict concrete compressive strength based on carefully selected attributes in the dataset.

Core Concepts:

TP (True Positives): Number of correctly predicted positive samples.

TN (True Negatives): Number of correctly predicted negative samples.

FP (False Positives): Number of incorrectly predicted positive samples.

FN (False Negatives): Number of incorrectly predicted negative samples.

TPR (True Positive Rate): The fraction of positive examples predicted correctly by the model.
Formula: TP/(TP + FN)

TNR (True Negative Rate): The fraction of negative examples predicted correctly by the model.
Formula: TN/(TN + FP)

FPR (False Positive Rate): The fraction of positive samples predicted incorrectly by the model.
Formula: FP/(TN + FP)

FNR (False Negative Rate): The fraction of negative samples predicted incorrectly by the model.
Formula: FN/(TP + FN)

Recall: The proportion of correctly predicted positives to all positives.
Formula: TP/(TP + FN)

Precision: The quality of positive predictions.
Formula: TP/(TP + FP)

F1-score: The measure of a model's accuracy, combining precision and recall.
Formula: (2 * TP)/(2 * TP + FP + FN)

Accuracy: The percentage of correctly predicted samples compared to all predicted samples.
Formula: (TP + TN)/(TP + FP + FN + TN)

Error Rate: The percentage of incorrectly predicted samples compared to all predicted samples.
Formula: (FP + FN)/(TP + FP + FN + TN)

BACC: The average of the true positive rate and true negative rate.
Formula: (TPR + TNR) / 2

TSS: The difference between the recall and the probability of false detection.
Formula: TP / (TP + FN) - FP / (FP + TN)

HSS: The probability of a prediction over the total prediction outcomes.
Formula: 2 * (TP * TN - FP * FN) / ((TP + FN) * (FN + TN) + (TP + FP) * (FP + TN))

Project Workflow:

Necessary Imports: In order to run the project, a cell can be inserted at the top of the notebook that includes all the necessary libraries, like such:

```
pip install scikit-learn pandas numpy tensorflow
```

Database Building: The dataset was downloaded from:
https://archive.ics.uci.edu/dataset/165/concrete+compressive+strength

Helper Functions: Two helper functions were made to obtain the performance metrics to evaluate the three models on the data. Another function was made to obtain an average of all performance metrics across the 10 folds of cross validation.

Data Loading and Preprocessing: The dataset is extracted from its file from the current folder and stored as a pandas dataframe in order to be used by the algorithms.

Feature Selection: Not all attributes were used to predict the target variable. A correlation matrix was used to see each attribute's correlation with the target variable, and the features with the highest correlations were selected, as they would provide the most insights during training.

Base Dataset Creation: The features and target variable are created as X and y from the selected features in the previous step. Since this dataset's target variable is a number, the t the concrete compressive strength was separated into 2 classes, strong and weak, based on whether it was above or below the mean.

Model Evaluation: The method for evaluating the models was chosen to be 10-fold cross validation. At the beginning of each iteration, the training and test sets were created at the index based on the current fold. Then, the random forest, SVM, and LSTM models were initialized. Since the LSTM model requires the data to be in time steps, the data was reshaped for that model. Next, the models were trained, and predictions were made on the test data. Finally, a confusion matrix was obtained for each model, supplying the true/false positives/negatives, which were used to calculate the rest of the metrics.

Average Metrics Calculation: A list of each metrics list obtained for each model, in each iteration, was kept for the 10-fold cross validation process. Using these lists, an average value for each metric across all 10 folds was calculated to give insights on each model's average performance.

ROC Curve and AUC: The ROC (Receiver Operating Characteristic) curve, along with its AUC (Area Under the Curve) is a metric to evaluate a model's ability to predict positive and negative classes for binary classification. This is outputted for all 3 models.

Deciding the Best Algorithm: If the ROC curve matches that of a right angle with its corner at the top left, the model is perfect, meaning that it can correctly predict the 2 classes. Of course, this is impossible, so the model with the ROC closest to this description, along with other attributes, is selected to be the best predictor of concrete compressive strength. The **Random Forest** model seems to be the best predictor of concrete compressive strength, due to its high AUC. It also performed the best out of the three models on average for nearly every metric.

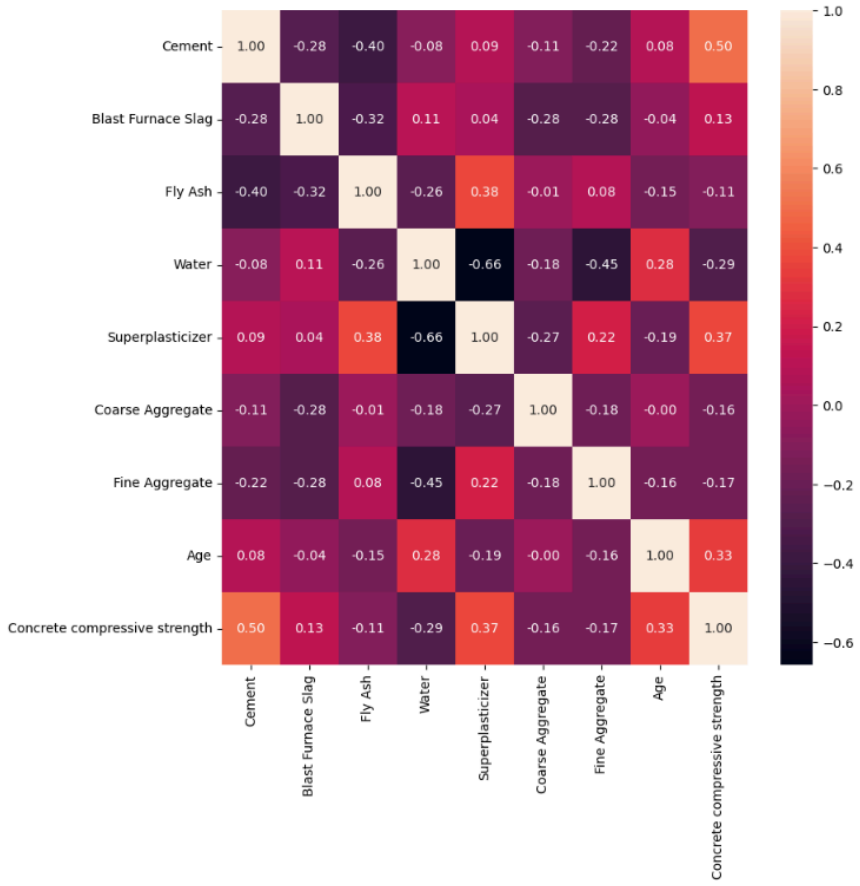
Conclusion:

In conclusion, this project demonstrates the application of several data mining techniques and concepts by implementing the Random Forest, SVM, and LSTM models to determine their efficiency and usage for predicting concrete compressive strength based on various attributes. The Random Forest model was found to be the best predictor for concrete compressive strength.

Screenshots

The correlation matrix for the data is shown below:

```
In [117]: correlation_matrix = comp_strength.corr()

fig, axis = plt.subplots(figsize=(9, 9))
sns.heatmap(correlation_matrix, annot=True, fmt='.2f', ax=axis)
plt.show()
```



The outputs for cross validation for all 3 models for iteration 1 and 10 are shown below:

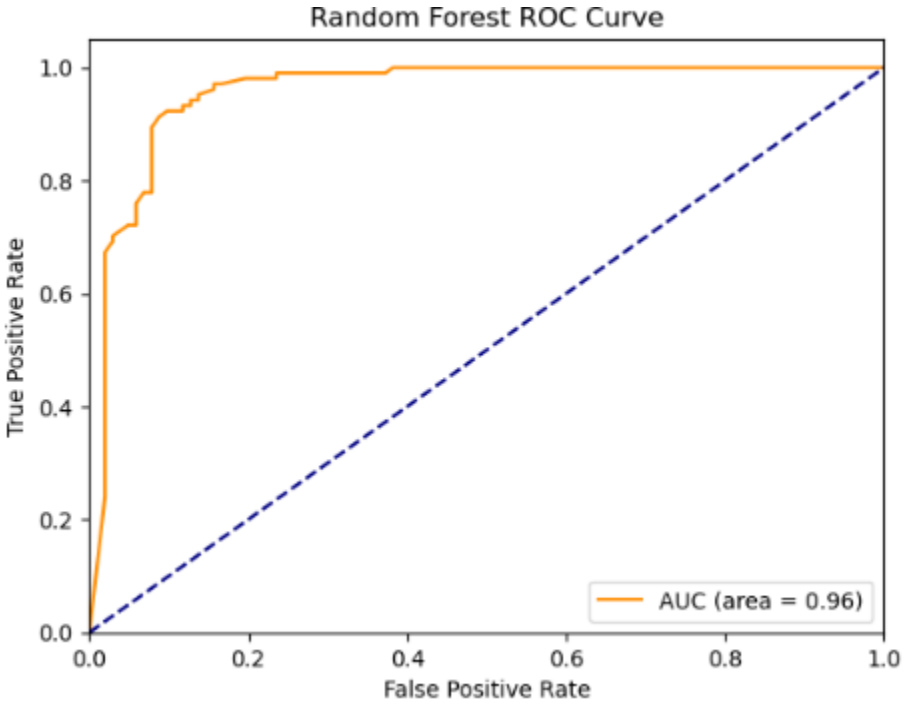
```
----- Metrics for all Algorithms in Iteration 1 -----

                RF    SVM   LSTM
TP           44.00  36.00  32.00
TN           47.00  46.00  48.00
FP            6.00   7.00   5.00
FN            6.00  14.00  18.00
TPR           0.88   0.72   0.64
TNR           0.89   0.87   0.91
FPR           0.11   0.13   0.09
FNR           0.12   0.28   0.36
Recall        0.88   0.72   0.64
Precision     0.88   0.84   0.86
F1-score      0.88   0.77   0.74
Accuracy      0.88   0.80   0.78
Error_rate    0.12   0.20   0.22
BACC          0.88   0.79   0.77
TSS           0.77   0.59   0.55
HSS           0.77   0.59   0.55
```

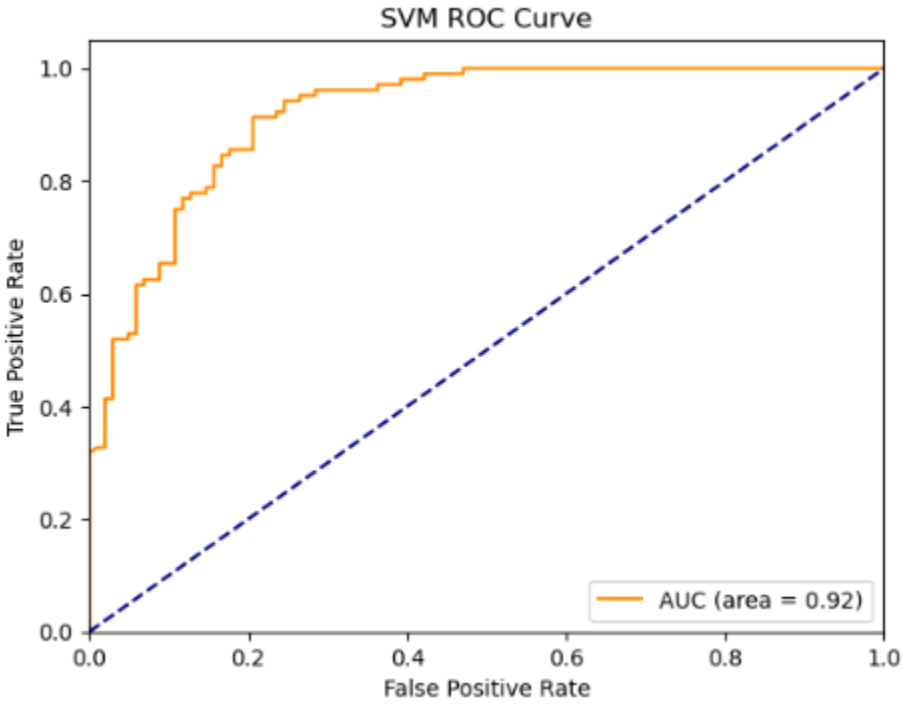
```
----- Metrics for all Algorithms in Iteration 10 -----

                RF    SVM   LSTM
TP           47.00  46.00  37.00
TN           48.00  43.00  43.00
FP            3.00   8.00   8.00
FN            5.00   6.00  15.00
TPR           0.90   0.88   0.71
TNR           0.94   0.84   0.84
FPR           0.06   0.16   0.16
FNR           0.10   0.12   0.29
Recall        0.90   0.88   0.71
Precision     0.94   0.85   0.82
F1-score      0.92   0.87   0.76
Accuracy      0.92   0.86   0.78
Error_rate    0.08   0.14   0.22
BACC          0.92   0.86   0.78
TSS           0.85   0.73   0.55
HSS           0.84   0.73   0.55
```

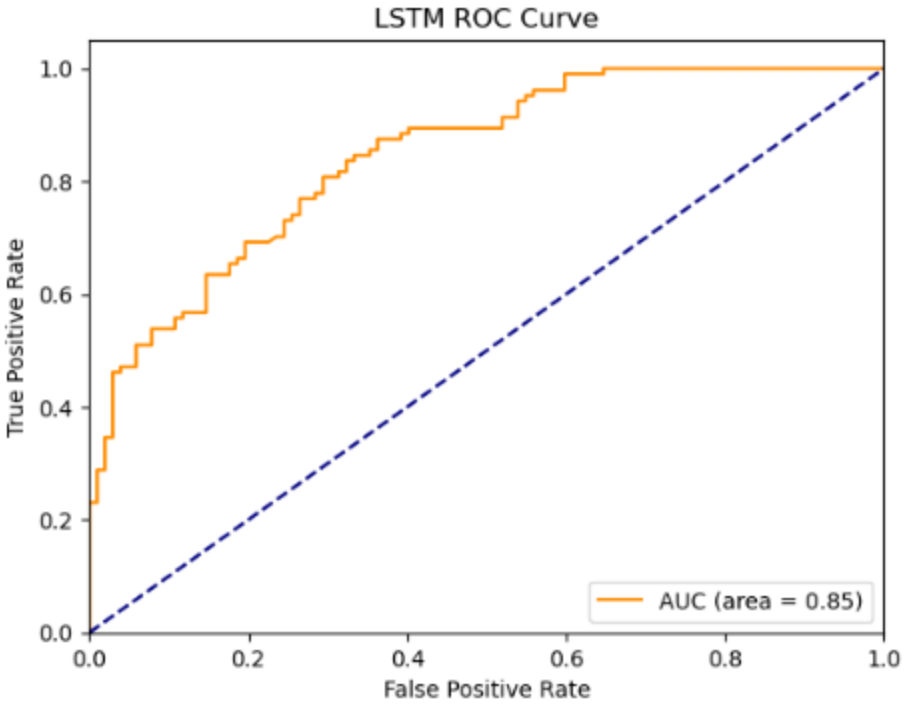
The output for the average performance metrics for all 3 models across all 10 folds is shown below:

```
----- Average Metrics for all Algorithms -----

                RF    SVM   LSTM
TP           44.60  40.20  37.60
TN           47.00  44.20  38.30
FP            5.30   8.10  14.00
FN            6.10  10.50  13.10
TPR           0.88   0.79   0.74
TNR           0.90   0.85   0.74
FPR           0.10   0.15   0.26
FNR           0.12   0.21   0.26
Recall        0.88   0.79   0.74
Precision     0.90   0.83   0.74
F1-score      0.89   0.81   0.73
Accuracy      0.89   0.82   0.74
Error_rate    0.11   0.18   0.26
BACC          0.89   0.82   0.74
TSS           0.78   0.64   0.48
HSS           0.78   0.64   0.47
```

The ROC and AUC curve and values for the Random Forest, SVM, and LSTM models respectively are shown below:






LSTM ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC (area = 0.85)

Other

The source code, notebook and the database file are all included in this zip file.

Link to Github Repository: https://github.com/GPintoNJIT/Concrete_Strength_Mining