Global Journal of Computer Science and Technology: D
Neural & Artificial Intelligence
Volume 24 Issue 1 Version 1.0 Year 2024
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Prediction of Hard Drive Failure using Machine Learning

By Elizabeth Atekoja

Abstract- The reliability of hard drives is paramount for maintaining data integrity and availability in cloud services and enterprise-level data centers where unexpected failures significantly impact operational efficiency and general performance. This work aims to develop a predictive model using regression analysis to accurately forecast imminent hard drive failures based on historical operational data, specifically SMART (Self-Monitoring Analysis and Reporting Technology) attributes. The study evaluated various regression models which comprises Decision Tree, Random Forest, Support Vector Machine (SVM), Gradient Boosting, and Neural Network. The outcomes indicated that the Random Forest model, with an MSE of 24.7427 and an R2 of 0.9876 and the Neural Network model, with an MSE of 22.6011 and an R2 of 0.7442, as the best performing models as they demonstrated high predictive accuracy and robustness. In contrast, the SVM model showed poor performance with an MSE of 2888.8623 and a negative R2 of -0.4465. Based on these outcomes, the Random Forest and Neural Network models are recommended for predicting hard drive failures as they delivered a balance of accuracy and interpretability.

GJCST-D Classification: LCC Code: QA76.9.D343

Prediction of Hard Drive Failure using Machine Learning

Elizabeth Atekoja

Abstract- The reliability of hard drives is paramount for maintaining data integrity and availability in cloud services and enterprise-level data centers where unexpected failures significantly impact operational efficiency and general performance. This work aims to develop a predictive model using regression analysis to accurately forecast imminent hard drive failures based on historical operational data, specifically SMART (Self-Monitoring Analysis and Reporting Technology) attributes. The study evaluated various regression models which comprises Decision Tree, Random Forest, Support Vector Machine (SVM), Gradient Boosting, and Neural Network. The outcomes indicated that the Random Forest model, with an MSE of 24.7427 and an R^2 of 0.9876 and the Neural Network model, with an MSE of 22.6011 and an R^2 of 0.7442, as the best performing models as they demonstrated high predictive accuracy and robustness. In contrast, the SVM model showed poor performance with an MSE of 2888.8623 and a negative R^2 of -0.4465. Based on these outcomes, the Random Forest and Neural Network models are recommended for predicting hard drive failures as they delivered a balance of accuracy and interpretability.

I. Background

The reliability of hard drives remains a vital concern across various sectors with particular significance in cloud services and enterprise-level data management where the integrity and availability of data are paramount. As businesses increasingly rely on data-driven decision-making, the cost of data loss or system downtime due to hard drive failures is substantial and impact operational efficiency, customer satisfaction, and financial performance. At enterprise-Level data centers, hard drives often operate under high-demand conditions, which can accelerate wear and tear. The data stored in these centers is crucial for operations and often includes transaction histories, client data, and business analytics. The failure of hard drives in such setups not only leads to data loss but also affects the redundancy and resilience of the entire data system. Regular monitoring and predictive maintenance facilitated by machine learning can significantly reduce the risk of such occurrences (Wang et al., 2018). Cloud services rely on data centers spread across multiple locations, where data is stored redundantly on numerous hard drives. Cloud providers must ensure high reliability to maintain service level agreements and customer trust. Predicting hard drive failure within these systems is not just a maintenance task but a critical operation that directly influences service quality and business continuity (Barroso et al., 2016). The implementation of predictive maintenance strategies for hard drives using machine learning algorithms has been shown to reduce unexpected downtime significantly. By analyzing SMART (Self-Monitoring, Analysis, and Reporting Technology) data, models can predict failures before they happen, allowing for timely replacements and repairs, thus minimizing downtime and reducing maintenance costs (Vishwanath and Nagappan, 2016).

Author: e-mail: elizabeth.atekoja@gmail.com

The study by Pinheiro, Weber and Barroso (2007) analyse a large population of disk drives and identifies common failure trends and indicators using statistical analysis to explore various SMART attributes. Critical indicators of failures were identified, but the study primarily focused on descriptive statistics rather than predictive modeling. Botezatu et al. (2016) developed prediction system for disk replacements in data centers to improve reliability. Machine learning models are employed to forecast when disks will need to be replaced, using a combination of real-time and historical SMART data. Achieved high accuracy in predictions; however, the study focused majorly on replacement timing rather than immediate failure detection. Hamerly and Elkan (2021) applied Bayesian methods to predict disk drive failures. The study uses Bayesian networks to model the probability of drive failures based on SMART attributes. In comparison, their method provides a probabilistic approach to prediction which is insightful for uncertainty estimation but did not align with the needs for precise regression predictions. With existing works mostly with statistical analysis and machine learning classification, there is a potential gap in exploring more comprehensive array of regression techniques, including advanced polynomial and non-linear regression models that might capture more complex relationships in the data.

Research Question

Which regression model provides the best performance in predicting hard drive failures based on the identified predictors?

II. Literature Review

Predicting failures is vital aspect of maintenance strategies which target is to prevent unplanned downtimes (Leukel, Gonzalez and Riekert, 2021). Concerning hard drive, while it is a resource that has been commonly adopted as a major storage device due to its cheap price and stability, rapid expansion of data storage systems expose it to failure (Gers, Schmidhuber and Cummins, 2000; Manousakis et al. 2016). According to Bairavasundaram et al. (2008), as disk capacity increases, chances of errors and data loss becomes high. By all standard, failure is costly and it is necessary they are detected or predicted (Murray, Hughes and Kreutz-Delgallo, 2005). Historical dataset based on SMART convention has aided utilization of machine learning in detection and prediction framework. Self-Monitoring and Reporting Technology (SMART) system uses attributes retrieved when hard drive perform and also during off-line tests to fix a failure prediction flag (Murray, Hughes and Kreutz-Delgallo, 2005). In most case, the SMART attributes in hard drive historical data is enormous while possessing target variable that enable classification (failure detection: yes or no) and regression (temporal prediction). Even when the data is primarily available and embedded with continuous and temporal attributes, most research in this domain focuses more on classification. Perhaps as it is good to detect the occurrence of failure, it is essential to predict when it is likely going to happen. A quick and random search on Google Scholar produced predictions based on classification (Table 1)

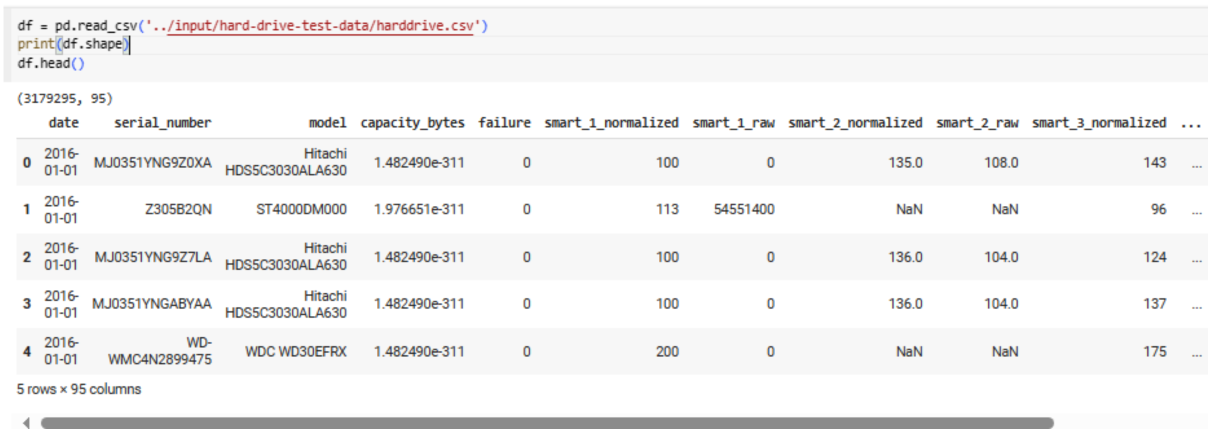
Table 1: Existing works on Hard Drive Prediction

Author	Problem Formation	Machine Learning Techniques
Aussel et al. 2017	Classification	SVM, Random Forest and Gradient Boosting
Shen et al. 2018	Classification	Random Forest
Garcia et al. 2018	Classification	Naïve Baye, SVM and Neural Network
Amran et al. 2021	Classification	Optimal Survival Trees and Optimal Classification Trees
Chhetri et al. 2022	Classification	Relational Graph Convolutional Neural Network
Ahmed and Green 2022	Classification	Random Forest, GBM and Logistic Regression
Wang et al. 2023	Classification	Naïve Baye, Random Forest, SVM, Gradient Boosted Decision Tree, CNN and LSTM
Zhang et al. 2023	Classification	SVM, Random Forest, Gradient Boosted Decision Tree and LSTM
Gour and Waoo 2023	Classification	LightGBM, Random Forest, Decision Tree, Deep Neural Network, Convolutional Neural Network and Recurrent Neural Network

Unlike many other investigations adoption classification problem, Anantharaman, Qiao and Jadav (2018) uses regression techniques to estimate the remaining useful life of hard disk drives directly. This contrasts with typical predictive models that classify whether a hard disk will fail within a specific timeframe. Random Forest and LSTM were used to analyse SMART attributes with focus of capturing historical temporal patterns that signify deterioration over time. The study evaluates the models based on their Mean Absolute Error (MAE), standard metric for regression tasks. While confidence level of models (R^2) were not reported in the work, Random Forest model achieved an MAE of 22.66 without a sliding window and 24.08 with a sliding window of size 10. For the LSTM models, the Many-to-One configuration recorded MAEs of 27.62 without a sliding window and 24.81 with it, at the same time the Many-to-Many setup had MAEs of 23.26 and 29.04 for the respective window configurations. Notably, the Piece-wise RUL approach in the Many-to-Many model significantly improved performance, yielding MAEs of 8.15 without a sliding window and 9.31 with a sliding window of size 10. Research by Zufle et al. (2020) applied both classification and regression to the prediction of hard drive failure using random forest machine learning techniques. Just like Anantharaman, Qiao and Jadav (2018), the confidence level metrics were not explicitly presented; the unfiltered data achieved a Mean Absolute Error (MAE) of 10 hours and a Root Mean Square Error (RMSE) of 44.6 hours and the pre-filtered data significantly improved these metrics, with an MAE of 4.5 hours and an RMSE of 12.8 hours. To explore more insights based on temporal and continuous-based predictions, this research will apply four machine learning regression including random forest, decision tree, SVM, Gradient Boosting and Neural Network.

III. Dataset Collection

Based on the view of Ruggiano and Perry (2019), secondary data can be collected and used to address new questions. A suitable and relevant dataset to achieve the study aim and objectives is sourced from Kaggle Hard Drive Test Data (kaggle.com). The dataset originated from BackBlaze and it is based on SMART statistics which makes it readily available for computational handling.



```
df = pd.read_csv('../input/hard-drive-test-data/harddrive.csv')
print(df.shape)
df.head()
```

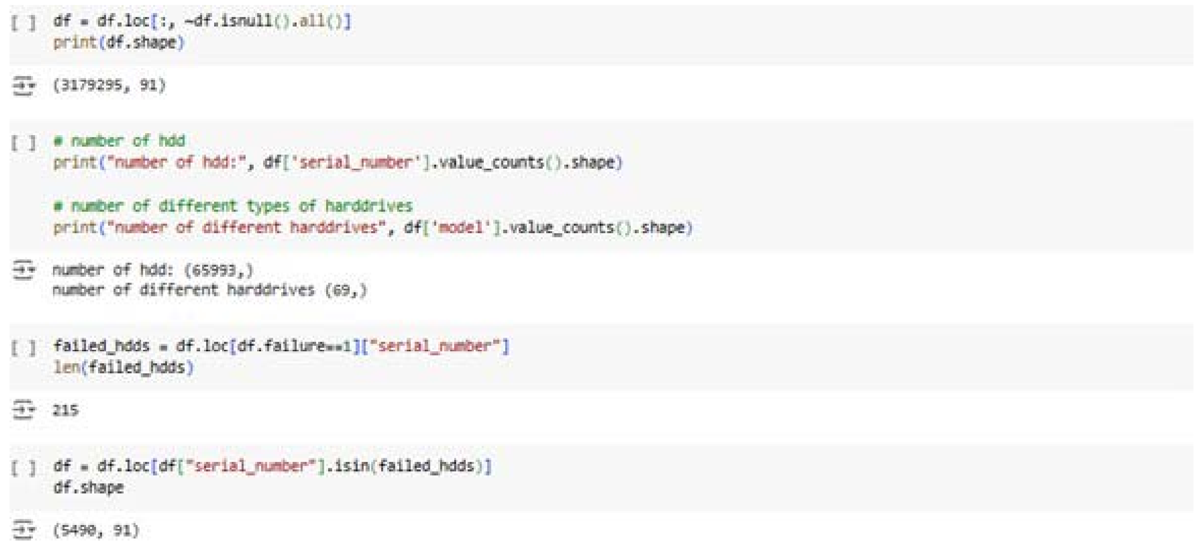
```
(3179295, 95)
```

	date	serial_number	model	capacity_bytes	failure	smart_1_normalized	smart_1_raw	smart_2_normalized	smart_2_raw	smart_3_normalized	...
0	2016-01-01	MJ0351YNG9Z0XA	Hitachi HDS5C3030ALA630	1.482490e-311	0	100	0	135.0	108.0	143	...
1	2016-01-01	Z305B2QN	ST4000DM000	1.976651e-311	0	113	54551400	NaN	NaN	96	...
2	2016-01-01	MJ0351YNG9Z7LA	Hitachi HDS5C3030ALA630	1.482490e-311	0	100	0	136.0	104.0	124	...
3	2016-01-01	MJ0351YNGABYAA	Hitachi HDS5C3030ALA630	1.482490e-311	0	100	0	136.0	104.0	137	...
4	2016-01-01	WD-WMC4N2899475	WDC WD30EFRX	1.482490e-311	0	200	0	NaN	NaN	175	...

5 rows × 95 columns

Figure 1: Extract of Hard Drive Performance Dataset

The dataset (Figure 1) contain data related to hard drive diagnostics with various attributes and readings from S.M.A.R.T (Self-Monitoring, Analysis, and Reporting Technology) data. The observation date is measure with attribute named "date". The "serial number" is the unique identifier for each hard drive, while the "model" denotes the model number of the hard drive. The drive's capacity is represented by "capacity_bytes", which is the size of the storage. The attribute "failure" is a binary indicator (0 or 1) where '1' indicates that the hard drive failed on this date. This attribute is used in prediction based on classification as the target variable. The remaining ninety attributes (columns) represent normalized and raw values of different SMART attributes which measure the health and performance of the hard drive. The preprocessing script (Figure 2) cleans the data by removing empty columns, identifies unique drives and models, isolates the subset of data concerning failed hard drives and provides basic counts that help understand the composition of the dataset.



```
df = df.loc[:, ~df.isnull().all()]
print(df.shape)
```

```
(3179295, 91)
```

```
# number of hdd
print("number of hdd:", df['serial_number'].value_counts().shape)

# number of different types of harddrives
print("number of different harddrives", df['model'].value_counts().shape)
```

```
number of hdd: (65993,)
number of different harddrives (69,)
```

```
failed_hdds = df.loc[df.failure==1]["serial_number"]
len(failed_hdds)
```

```
215
```

```
df = df.loc[df["serial_number"].isin(failed_hdds)]
df.shape
```

```
(5490, 91)
```

Figure 2: Extract of Data Pre-Processing

IV. Model Development and Implementation

a) Decision Tree

Decision Tree is executed using the *DecisionTreeRegression* function (Figure 3). The function builds a regression tree by recursively splitting the dataset based on feature values to minimize variance within the target variable which ensure each split results in subsets with reduced variance compared to the parent node. The implementation starts by confirming that the dataset can be split further based on the minimum sample size or maximum tree depth; if not, it returns the mean target value of the dataset as a leaf node. The function iterates over all possible splits for each feature, computes the variance reduction for each and selects the split that maximizes this reduction. When the best split is achieved, the dataset is divided and the function is called recursively for each subset until leaf nodes are formed.

```
Function Decision_Tree_Regression(Data, Depth, Max_Depth, Min_Split_Size)
  // Input: Data - the dataset for building the regression tree
  //        Depth - the current depth of the tree
  //        Max_Depth - the maximum allowable depth for the tree
  //        Min_Split_Size - the minimum size of the dataset to allow further splits

1. If Data has fewer rows than Min_Split_Size or Depth equals Max_Depth:
  2. Compute and return the mean of the target variable in Data (create a leaf node)

3. Initialize best_score to infinity // Used to track the best variance reduction found
4. Initialize best_split to null // Stores the best feature and split point

5. For each feature in Data:
  6. For each possible split point within this feature:
    7. Split Data into two subsets (Data_Left and Data_Right) based on this split point
    8. Calculate the sum of squared residuals from the mean in both subsets
    9. Compute the variance reduction as the difference between the variance before and after the split
    10. If the variance reduction is greater than best_score:
      11. Update best_score with this variance reduction
      12. Update best_split with this feature and split point

13. If best_split is null: // No valid split was found
  14. Compute and return the mean of the target variable in Data (create a leaf node)

15. Use best_split to partition Data into Data_Left and Data_Right

16. Create a node that stores the feature and split point from best_split

17. Recursively apply Decision_Tree_Regression to Data_Left:
  18. Left_Child = Decision_Tree_Regression(Data_Left, Depth + 1, Max_Depth, Min_Split_Size)

19. Recursively apply Decision_Tree_Regression to Data_Right:
  20. Right_Child = Decision_Tree_Regression(Data_Right, Depth + 1, Max_Depth, Min_Split_Size)

21. Attach Left_Child and Right_Child to the current node as its branches

22. Return the current node // This node now represents the decision at this level of the tree
End Function
```

Figure 3: Pseudocode for Implementing Decision Tree

The model by the decision tree technique provides a strong alignment between actual and predicted values along the central line but it exhibits noticeable variance and outliers (Figure 4). This indicates that while the Decision Tree will predict hard drive failures using the SMART dataset while its predictions are less stable and accurate with more frequent errors.

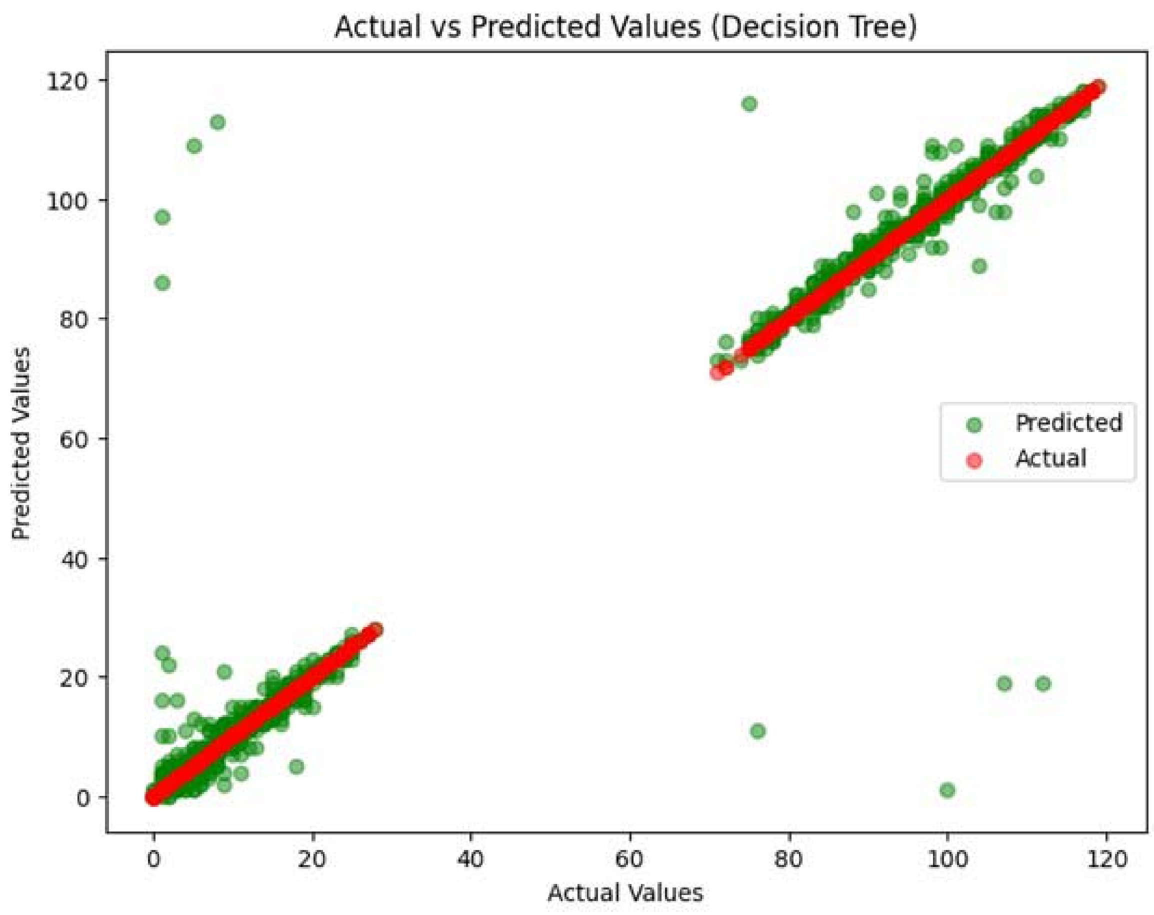


Figure 4: Decision Tree Prediction Outcome

b) Random Forest

The algorithm is implemented by creating an empty list, *forests*, to store the decision trees (Figure 5). For each tree in the list *forest*, it generates a bootstrapped sample of the training data. It fits a decision tree regression model using specified parameters like the number of features to consider for each split, the minimum number of samples required to split a node and the maximum depth of each tree. These trees are appended to the forests list. To make predictions, the *predict* function aggregates predictions from all trees for the test data, averaging their outputs to produce the final prediction.

```
Inputs:
   data: training dataset
n_trees: number of trees in the forest
n_features: number of features to consider for each split
min_samples_split: minimum number of samples required to split a node
max_depth: maximum depth of each tree
Begin
   forests = []
   for i from 1 to n_trees do
bootstrapped_data = bootstrap_sample(data)
      tree = DecisionTreeRegression(bootstrapped_data, n_features, min_samples_split, max_depth)
forests.append(tree)
   end for
   function predict(test_data)
      predictions = []
      for tree in forests do
```

```
predictions.append(tree.predict(test_data))
        end for
final_prediction = average(predictions)
        return final_prediction
    end function
End
```

Figure 5: Pseudocode for Implementing Random Forest

The outcome of random forest showed model that is highly effective at predicting hard drive failures using the SMART dataset (Figure 6). However, there are some deviations particularly at higher value ranges and this indicate instances where the model's predictions are less accurate.

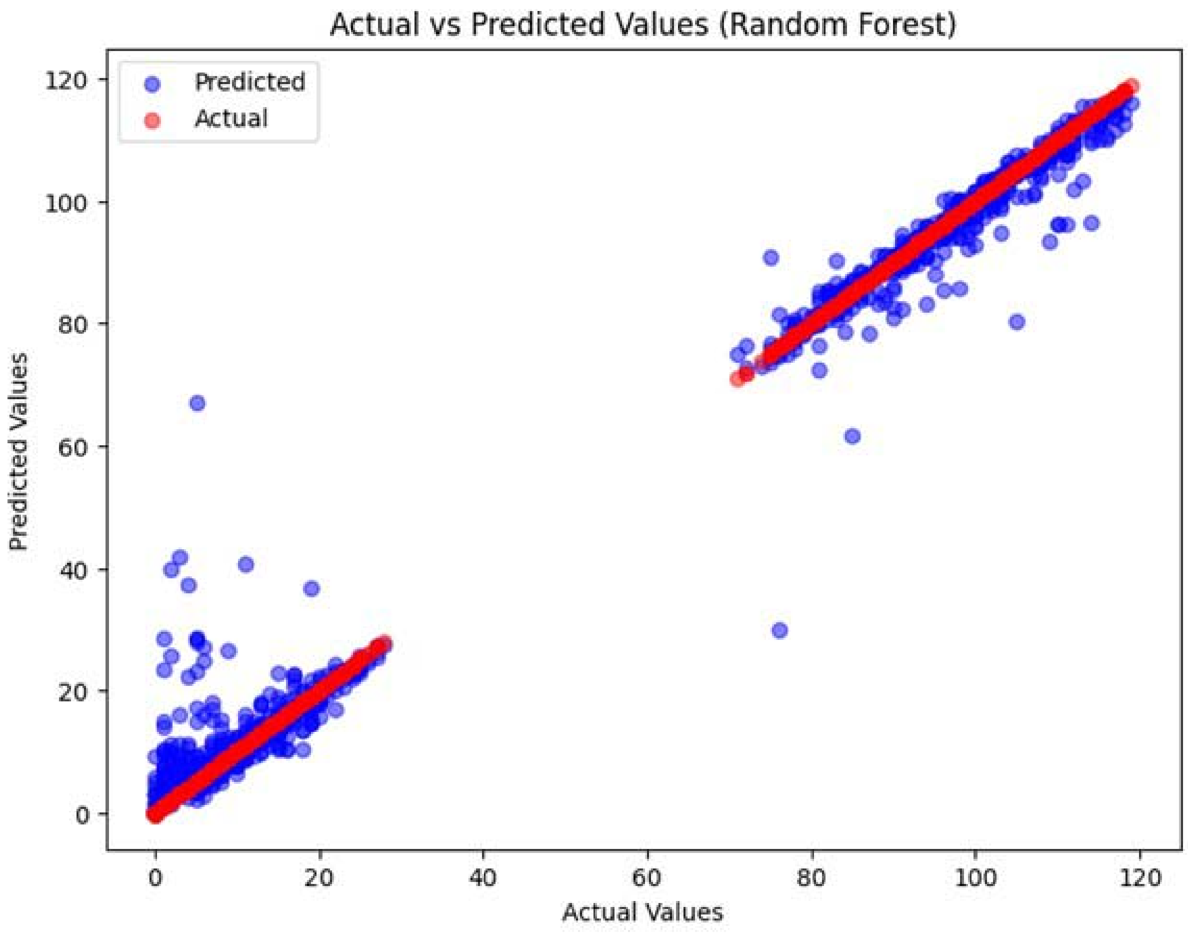


Figure 6: Random Forest Prediction Outcome

c) *Support Vector Machine (SVM)*

From Figure (7.), the implementation begins by initializing the dataset by specifying parameters like epsilon (for error tolerance), the regularization parameter (C), and the kernel type (linear, polynomial, or radial basis function). The kernel function is defined based on the selected kernel type which transforms the input data into a higher-dimensional space to enable separation. For each data point in the training set, the loss function is computed considering the hinge loss for points outside the epsilon margin and at the same time the model parameters are optimized using an algorithm like Sequential Minimal Optimization. The training continues iteratively until convergence criteria are met, after which the final model parameters are saved to enable the SVM to predict hard drive failures on new instances by computing their positions relative to the support vectors and summing the weighted contributions.

```
1. Input:
    - Data: Dataset containing features and target values
    - Epsilon: Specifies the epsilon-tube within which no penalty is associated in the training loss
function with points predicted within a distance epsilon from the actual value
    - C: Regularization parameter, which defines the trade-off between achieving a low error on the
training data and minimizing model complexity for better generalization
    - Kernel_Type: Type of kernel function to use ( linear, polynomial, radial basis function)

2. Initialize the model parameters (weights and bias) to zero or small random values

3. Define the kernel function based on Kernel_Type:
    - If linear: Use the linear dot product
    - If polynomial: Use (gamma * dot_product + coef0)^degree
    - If radial basis function (RBF): Use exp(-gamma * |a-b|^2)

4. For each instance in the training Data:
    - Calculate the loss function considering:
        - Hinge loss for points outside the epsilon margin
        - Regularization term using C

5. Use an optimization algorithm (e.g., Sequential Minimal Optimization) to:
    - Select a subset of training instances as support vectors
    - Optimize the model parameters to minimize the objective function (loss + regularization)

6. Continue iterating over the training set until convergence criteria are met, such as:
    - No substantial change in the loss function
    - Reaching a maximum number of iterations

7. Save the model parameters (support vectors, weights, bias) after training is complete

Testing:

1. Input:
    - Model: The trained SVM regression model (containing support vectors, weights, bias)
    - New_Data: New instances for which to predict the target values

2. For each instance in New_Data:
    - Compute the prediction by applying the kernel function between the new instance and each
support vector, scaled by the corresponding weight, and summed with the bias

3. Return the predictions for all instances in New_Data
```

Figure 7: Pseudocode for Implementing SVM

The outcome (Figure 8) indicates that the Support Vector Machine (SVM) model struggles with accurately predicting hard drive failures as evidenced by the clustering of predicted values around specific ranges and significant deviations from the actual values.

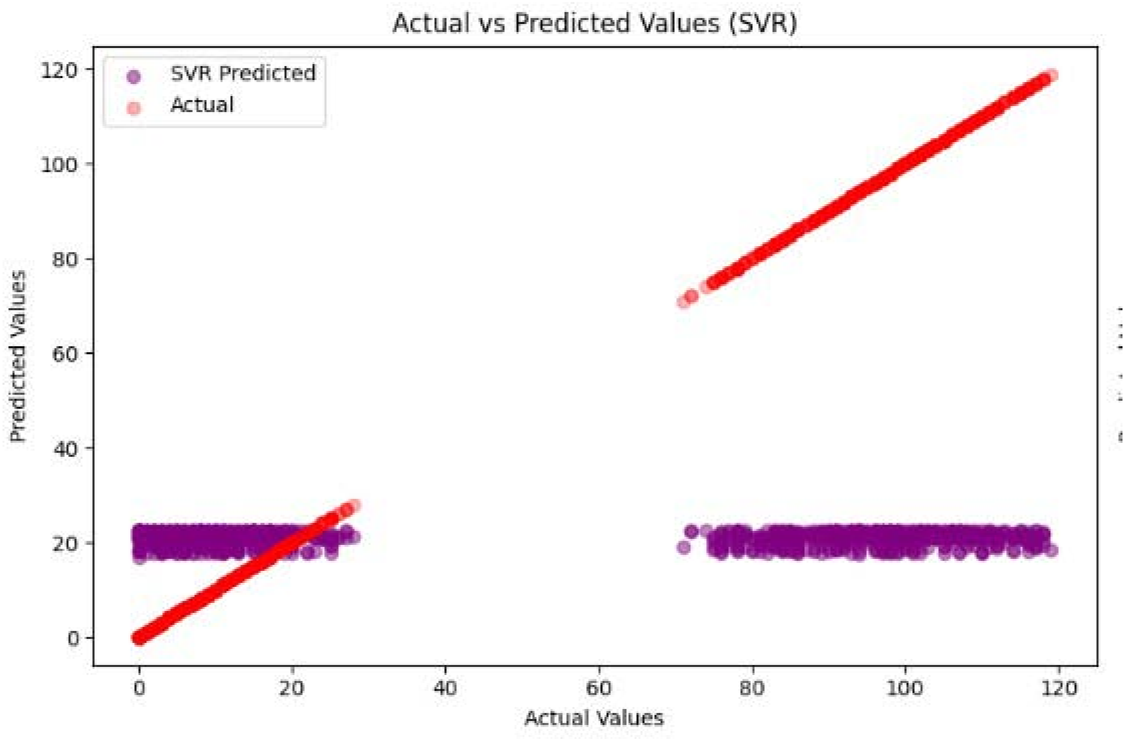


Figure 8: SVM Prediction Outcome

d) Gradient Boosting

The implementation of Gradient Boosting starts with initializing the model, which initially predicts the mean of the target values (Figure 9). For each of the specified number of trees, the model computes the residuals (differences between actual target values and current predictions), which serve as the target for fitting the next tree. Each new tree is trained on these residuals, adhering to constraints like maximum depth and minimum samples required to split. The model is updated by adding the scaled predictions of the new tree to the current model's predictions, iteratively improving accuracy. This process continues until all trees are built, making an ensemble model that makes final predictions on new data by aggregating the contributions of all trees.

```
1. Input:
   - Data: Dataset containing features and target values
   - Learning_Rate: Shrinks the contribution of each tree by this factor to improve model robustness
   - N_Trees: Number of trees to build (iterations)
   - Max_Depth: Maximum depth of each tree
   - Min_Samples_Split: Minimum number of samples required to split an internal node

2. Initialize:
   - Model = an initial model which could just predict the mean of the target values of Data

3. For i = 1 to N_Trees:
   4. Compute the residuals (negative gradient of the loss function) for each training instance:
      - Residuals = actual target values - predictions from the current Model

   5. Fit a new tree to the residuals using the feature values with the constraints of Max_Depth and
Min_Samples_Split

   6. Predict the residuals for the training dataset using this new tree

   7. Update the Model by adding the scaled predictions of the new tree:
        - Model = Model + Learning_Rate * new tree predictions

8. Return the final Model

Testing:

1. Input:
    - Model: The trained Gradient Boosting model (ensemble of trees)
    - New_Data: New instances for which to predict the target values

2. Start with the initial prediction (often the mean of the training target values if that was the
initialization)

3. For each tree in the Model:
    4. Update the prediction for each instance in New_Data:
        - Prediction += Learning_Rate * tree's prediction on New_Data

5. Return the final predictions for all instances in New_Data
```

Figure 9: Pseudocode for Implementing Gradient Boosting

The results (Figure 10) indicate that the Gradient Boosting model produces predictions that align closely with the actual values, as seen by the clustering of prediction dots around the line of equality. However, there is a noticeable dispersion in the predicted values, particularly at the lower and middle ranges.

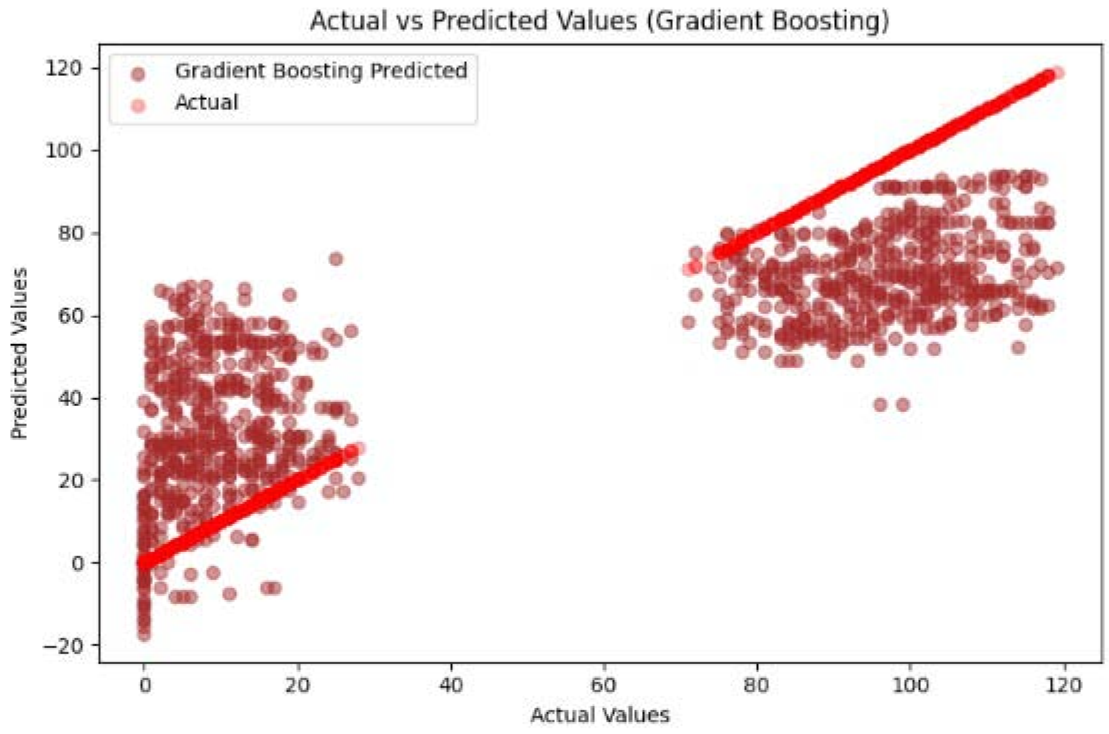


Figure 10: SVM Prediction Outcome

e) Neural Network

The implementation (Figure 11) starts with defining the network's structure and this includes the number of input neurons (corresponding to the dataset features), hidden layers and output neurons. The network is initialized with random weights and biases for each layer. During training, the network processes each training instance by propagating inputs through the layers with the application of activation functions at each hidden layer and computation of the output. The loss function which stands for mean squared error measures the difference between the predicted and actual values while backpropagation is used to update the weights and biases by minimizing this loss. After training, the neural network is tested by passing new data through the trained model to predict hard drive failures, with aggregating predictions for final output.

```
1. Input:
    - Num_Input_Neurons: Number of neurons in the input layer, corresponding to the number of features in the dataset
    - Hidden_Layers: Number of hidden layers in the network
    - Neurons_Per_Layer: Array containing the number of neurons in each hidden layer
    - Output_Neurons: Number of neurons in the output layer

2. Structure:
    - Create a network structure with the specified number of layers and neurons
    - Initialize weights and biases for each layer randomly

3. Return the initialized network with weights and biases

1. Input:
    - Network: Initialized neural network structure with weights and biases
    - Data: Training dataset features
    - Labels: Training dataset target values (continuous)
    - Learning_Rate: Step size for updating the weights
    - Epochs: Number of times to iterate over the entire training dataset

2. Training Process:
    - For each epoch:
        3. For each instance (Input, Target) in (Data, Labels):
            4. Forward_Propagation:
                - Compute the output of each layer starting from input to output layer
                - Use activation functions like ReLU for hidden layers and linear for the output layer

            5. Compute Loss:
                - Calculate the mean squared error or another suitable loss function between the predicted and actual values

            6. Backward_Propagation:
                - Calculate gradients of the loss function with respect to each weight and bias
                - Update weights and biases using these gradients and the learning

Testing:

1. Input:
    - Network: Trained neural network
    - New_Data: New instances for which to predict the target values

2. Prediction Process:
    - For each instance in New_Data:
        3. Forward_Propagation:
            - Compute the output using the trained network starting from the input layer to the output layer

4. Collect and return all predictions for New_Data
```

Figure 11: Pseudocode for Implementing Neural Network

The outcome (Figure 12) shows a significant reduction in both training and validation loss over the epochs, indicating that the neural network is learning and improving its predictions for hard drive failures using the SMART dataset. The close alignment between the training and validation loss curves suggests that the model generalizes well to unseen data, minimizing overfitting and ensuring robust predictive performance.

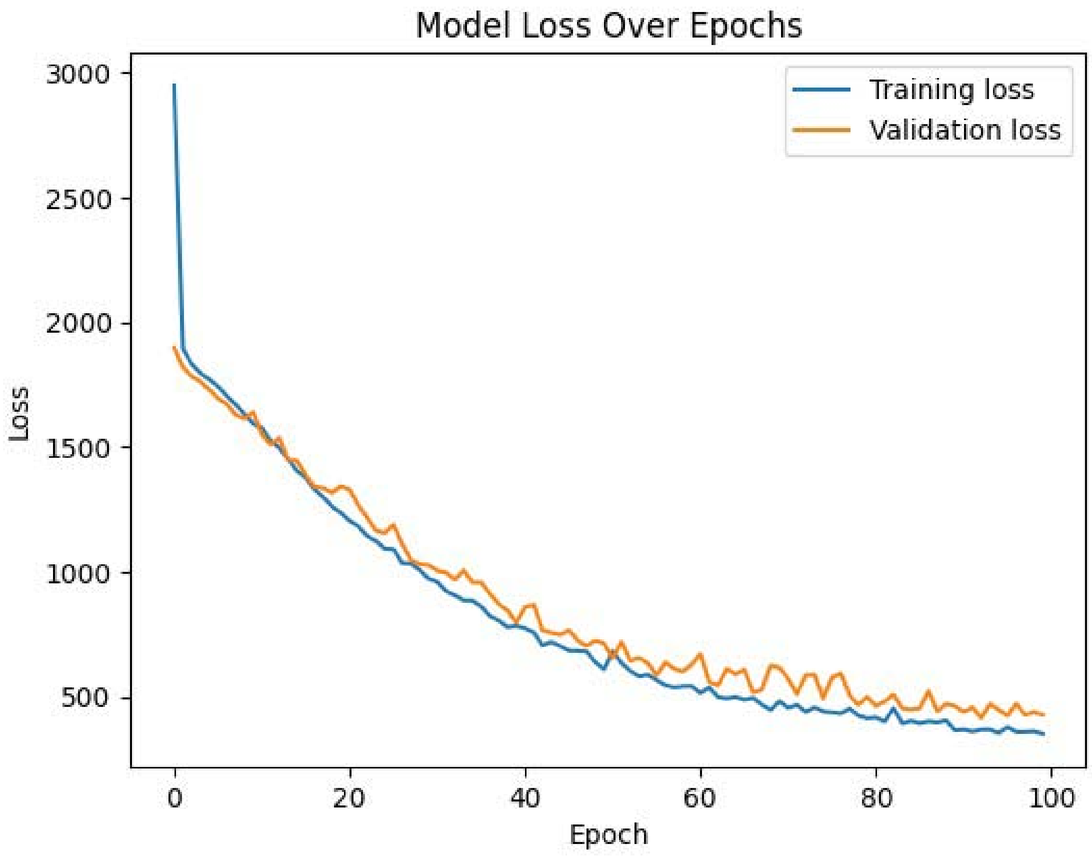


Figure 12: Neural Network Prediction Outcome

V. Results and Discussion

For the Decision Tree model (Table 2), the MSE is 71.4943, with an R^2 of 0.9642, indicating a high degree of accuracy. The Random Forest model has an MSE of 24.7427 and an R^2 of 0.9876, suggesting even better performance. The SVM model shows an MSE of 2888.8623 and an R2 of -0.4465, indicating poor predictive capability. The Gradient Boosting model has an MSE of 823.1132 and an R^2 of 0.5879, reflecting moderate accuracy. Lastly, the Neural Network model reports an MSE of 22.6011 and an R^2 of 0.7442, which indicates good performance.

Table 2: Models Results

Models	MSE	R^2
Decision Tree	71.4943	0.9642
Random Forest	24.7427	0.9876
SVM	2888.8623	-0.4465
Gradient Boosting	823.1132	0.5879
Neural Network	22.6011	0.74424

The Decision Tree model (Figures 13 and 14) exhibits a relatively low MSE of 71.4943, indicating minimal errors in predictions, and a high R^2 value of 0.9642, suggesting it explains a large portion of the variance in the target variable effectively. The Random Forest model performs even better, with a significantly lower MSE of 24.7427 and an R^2 of 0.9876, indicating excellent explanatory power. This shows that the ensemble approach of combining multiple decision trees improves prediction accuracy and robustness, reducing the impact of overfitting associated with individual trees.

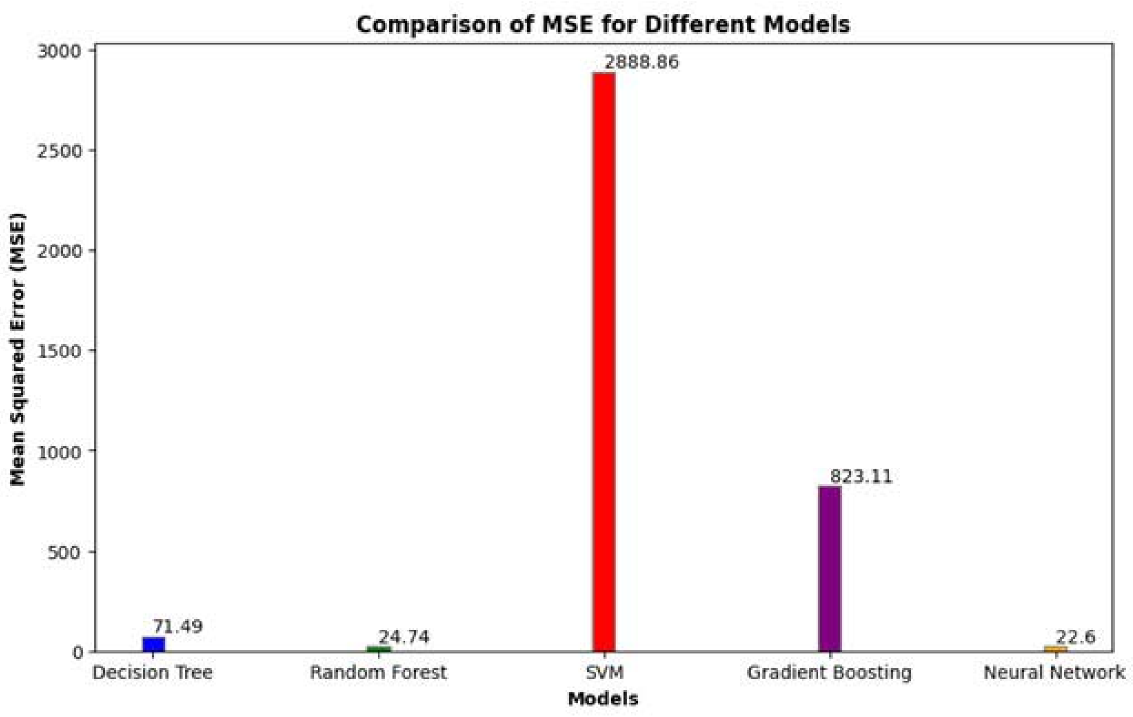


Figure 13: MSE for Selected Models

In contrast, the Support Vector Machine (SVM) model performs poorly with a very high MSE of 2888.8623 and a negative R^2 value of -0.4465, suggesting it fails to capture the relationship between the features and the target variable and is worse than a simple mean predictor. Gradient Boosting shows moderate performance with an MSE of 823.1132 and an R^2 of 0.5879, indicating more prediction errors and less variance explained compared to the Decision Tree and Random Forest models. While powerful, Gradient Boosting may require more fine-tuning or may not be as effective on the study's dataset.

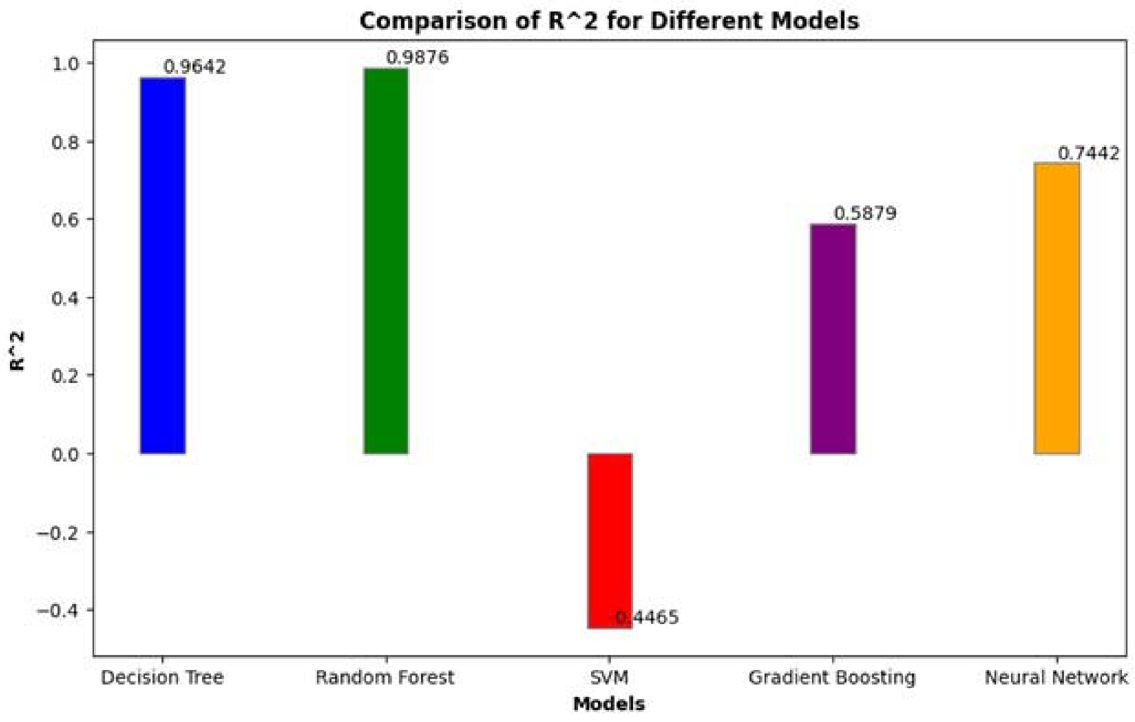


Figure 13: R^2 for Selected Models

The Neural Network model has the lowest MSE of 22.6011, indicating very accurate predictions with minimal errors, but its R^2 value of 0.7442, though high, is not as close to 1 as that of the Random Forest, suggesting some unexplained variance in the data. Neural networks can be very powerful but require substantial data and computational resources, and their performance is highly dependent on the architecture and training process. Overall, the Random Forest perform best with the lowest errors and highest explanatory power. The Neural Network also performs well with very low errors but slightly less explanatory power. The Decision Tree model demonstrates strong performance with good accuracy and explanatory power, while Gradient Boosting has moderate performance. The SVM is not suitable for this task based on the given metrics.

VI. Conclusion, Recommendation and Limitations

The research into predictive modeling for hard drive failures using various regression techniques has shown remarkable insights into the efficiency of different models in this domain. The Random Forest and Neural Network models emerged as the best predictors as they demonstrated the highest accuracy and predictive power. The Random Forest model, with an MSE of 24.7427 (minimal predicting error), and an R^2 of 0.9876 proved highly effective in capturing the complex relationships within the SMART dataset. Similarly, the Neural Network model exhibited strong predictive capabilities, with an MSE of 22.6011 and an R^2 of 0.7442, which indicates robust performance and generalizability. Based on the performance metrics, both random forest and neural network is recommended for predicting hard drive failures. Despite the promising results, a few limitations exist:

1. Model Interpretability: While Random Forest and Decision Tree models offer relatively high interpretability through feature importance, Neural Networks and Gradient Boosting model can be perceived as black-box models. Techniques like SHAP and LIME will help in the interpretability process and overall models explainability.
2. Generalizability: The models were trained and tested on a specific dataset. Their generalizability to other datasets or different types of hard drives may vary. Cross-validation and testing on diverse datasets are important to ensure robustness.
3. Model Maintenance: Predictive models need continuous monitoring and updating to maintain accuracy over time. Changes in hard drive technology and operational conditions necessitate periodic retraining of the models.
4. In conclusion, while the study demonstrates the effectiveness of advanced regression techniques in predicting hard drive failures, addressing these limitations through ongoing research and development is essential for improving model reliability and applicability in real-world scenarios.

References Références Referencias

1. Ahmed, J. and Green, R.C. (2022) Predicting severely imbalanced data disk drive failures with machine learning models. *Machine Learning with Applications* 9, 100361.
2. Amram, M. et al. (2021) Interpretable predictive maintenance for hard drives. *Machine Learning with Applications* 5, 100042.
3. Anantharaman, P., Qiao, M. and Jadav, D. (2018) Large Scale Predictive Analytics for Hard Disk Remaining Useful Life Estimation. *IEEE International Congress on Big Data (Big Data Congress), San Francisco, CA, USA*, 251-254. doi: 10.1109/Big DataCongress.2018.00044.
4. Aussel, N. et al. (2017) Predictive models of hard drive failures based on operational data. *ICMLA 2017: 16th IEEE International Conference on Machine Learning and Applications, Cancun, Mexico*, 619 - 625. doi: 10.1109/ICMLA.2017.00-92.
5. Bairavasundaram, L. N. et al. (2008) An analysis of data corruption in the storage stack. *ACM Transactions on Storage* 4, p. 8.
6. Botezatu, M. et al. (2016) Predicting Disk Replacement towards Reliable Data Centers. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM.
7. Chhetri, T.R. et al. (2022) Knowledge Graph Based Hard Drive Failure Prediction. *Sensors* 22, 985. doi: 10.3390/s22030985.
8. Garcia, M. et al. (2018) Review of techniques for predicting hard drive failure with SMART attributes. *International Journal of Machine Intelligence and Sensory Signal Processing* 2(2), 151-164.
9. Gers, F.A., Schmidhuber, J. and Cummins, F. (2000) Learning to forget: Continual prediction with LSTM. *Neural Computation* 12, 2451–2471.
10. Gour, L. and Waoo, A.A. (2023). Hard disk drive failure prediction in the data centre using ensemble learning with deep neural network model. *Journal of Southwest Jiaotong University* 58(1), 803-815. Available at: https://jsju.org.cn/

11. Hamerly, G. and Elkan, C. (2001) Bayesian Approaches to Failure Prediction for Disk Drives. *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williamstown, MA, USA, Morgan Kaufmann.*
12. Leukel, J., González, J. and Riekert, M. (2021) Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review. *Journal of Manufacturing Systems* 61, 87–96. doi: 10.1016/j.jmsy.2021.08.012.
13. Manousakis, I. (2016) Environmental conditions and disk reliability in free-cooled datacenters. *USENIX-The Advanced Computing Systems Association*. 7552ae55cbf6b440a45615585aeb621c.pdf (cloudworks.nu)
14. Murray, J.F., Hughes, G.F. and Kreutz-Delgado, K. (2018) Machine Learning Methods for Predicting Failures in Hard Drives: A Multiple-Instance Application. *Journal of Machine Learning Research* 6, 783–816.
15. Ruggiano, N. and Perry, T. E. (2019) Conducting secondary analysis of qualitative data: Should we, can we, and how? *Qualitative Social Work: Research and Practice 18*(1), 81–97. https://doi.org/ 10 1177/ 1473325017700701.
16. Shen, J. et al. (2018) Random-forest-based failure prediction for hard disk drive. *International Journal of Distributed Sensor Networks* 14(11). doi: 10.1177/1550147718806480.
17. Wang, H. (2023) Optimizing Efficiency of Machine Learning Based Hard Disk Failure Prediction by Two-Layer Classification-Based Feature Selection. *Applied Sciences* 13, 7544. doi: 10.3390/app13137 544.
18. Zhang, M. et al. (2023) Hard Disk Failure Prediction Based on Blending Ensemble Learning. *Applied Sciences* 13, 3288. doi: 10.3390/app13053288.
19. Züfle, M. et al. (2020) To Fail or Not to Fail: Predicting Hard Disk Drive Failure Time Windows', Measurement, Modeling and Evaluation of Computing Systems (MMB 2020*). Lecture Notes in Computer Science* 12040. Springer, Cham. doi: 10.1007/978-3-030-43024-5_2