**Results analysis**

**Steps:**

1. **Started by checking the data and search for nulls which there were none A screenshot of a computer

   Description automatically generated**
2. **Checked all the classes of the target column to see if there is a leaking feature which turned out to be the “Target” column that is = 1 in case of failures( except for random failuresA screenshot of a computer

   Description automatically generated**A screenshot of a computer

   Description automatically generated
3. **Viewed all classes statistics with respect to column to better understand the data**

**A screenshot of a computer

Description automatically generated**

1. **Then Encoded the failure type column with label encoding and the machine type with one hot encoding and joined them together into one df.** **A screenshot of a computer

   Description automatically generated**
2. **Then checked the Id column which turned out to be all unique , so I dropped it alongside UDI and dropped the “Target” column (leaking feature) from one DF and kept it in another one .**

**A white rectangular object with text

Description automatically generated**

1. **Then splitted both leaky and non-leaky DFs into 80% training and 20% testing & applied SMOTE to balance the data.**

**A screenshot of a computer program

Description automatically generated**

1. **Built 2 logistic regression models , one for the leaky data and one for the non-leaky data(before applying SMOTE).**

****

1. **Built 2 Naïve Bayes models , one for the leaky data and one for the non-leaky data (before applying SMOTE)..**

**A screenshot of a computer

Description automatically generated**

**Model performance comparison with and without leakage (before applying SMOTE):**

**Leaky Logistic Regression vs. Non-leaky Logistic Regression:**

|  |  |  |
| --- | --- | --- |
| **Accuracy** | **Leaky Logistic Regression** | **Non-leaky Logistic Regression** |
|  | **0. 987** | **0.977** |

The Leaky Logistic Regression model achieves higher accuracy (0.987) compared to the Non-leaky Logistic Regression model (0.977). Data leakage contributes to this higher accuracy by allowing the model to inadvertently learn patterns present in the test data.

|  |  |  |
| --- | --- | --- |
| **Precision** | **Leaky Logistic Regression** | **Non-leaky Logistic Regression** |
| **Heat Dissipation Failure** | **1.00** | **0.60** |
| **No Failure** | **0.99** | **0.98** |
| **Overstrain Failure** | **0.86** | **0.69** |
| **Power Failure** | **0.95** | **0.82** |
| **Random Failures** | **0.0** | **0.0** |
| **Tool Wear Failure** | **0.0** | **0.0** |

In general, the Leaky Logistic Regression model exhibits higher precision across most failure types compared to the Non-leaky Logistic Regression model. This indicates that the Leaky Logistic Regression model makesfewer false positive predictions, which could be a consequence of the model exploiting leakage to make overly confident predictions, but both failed to identify random failures due the 0 value in the “Target” column and the similar range of numbers in the meta data between random failures and other classes alongside tool wear which is weird but indicting possible class imbalance.

|  |  |  |
| --- | --- | --- |
| **Recall** | **Leaky Logistic Regression** | **Non-leaky Logistic Regression** |
| **Heat Dissipation Failure** | **0.60** | **0.20** |
| **No Failure** | **1.00** | **0.99** |
| **Overstrain Failure** | **0.92** | **0.69** |
| **Power Failure** | **0.95** | **0.90** |
| **Random Failures** | **0.0** | **0.0** |
| **Tool Wear Failure** | **0.0** | **0.0** |

The Leaky Logistic Regression model also demonstrates higher recall for certain failure types, indicating that it captures more true positive instances. Again because it already knows when the failure happens.

|  |  |  |
| --- | --- | --- |
| **F1-score** | **Leaky Logistic Regression** | **Non-leaky Logistic Regression** |
| **Weighted avg** | **0. 98** | **0.97** |

F1-score: The F1-score, which considers both precision and recall, is also higher for the Leaky Logistic Regression model (Weighted avg: 0.98) compared to the Non-leaky Logistic Regression model (Weighted avg: 0.97).

**Leaky Naive Bayes vs. Non-leaky Naive Bayes :**

|  |  |  |
| --- | --- | --- |
| **Accuracy** | **Leaky Naive Bayes** | **Non-leaky Naive Bayes** |
|  | **0. 991** | **0.947** |

Like logistic regression, the Leaky Naive Bayes model achieves higher accuracy (0.9745) compared to the Non-leaky Naive Bayes model (0.9185).

|  |  |  |
| --- | --- | --- |
| **Precision** | **Leaky Naive Bayes** | **Non-leaky Naive Bayes** |
| **Heat Dissipation Failure** | **0.70** | **0.32** |
| **No Failure** | **1.00** | **0.98** |
| **Overstrain Failure** | **0.75** | **0.30** |
| **Power Failure** | **0.87** | **0.21** |
| **Random Failures** | **0.0** | **0.0** |
| **Tool Wear Failure** | **1.00** | **0.0** |

The Leaky Naive Bayes model generally exhibits higher precision across failure types compared to the Non-leaky Naive Bayes model. This suggests that the Leaky Naive Bayes model makes fewer false positive predictions due to data leakage.

|  |  |  |
| --- | --- | --- |
| **Recall** | **Leaky Naive Bayes** | **Non-leaky Naive Bayes** |
| **Heat Dissipation Failure** | **1.00** | **0.67** |
| **No Failure** | **1.00** | **0.96** |
| **Overstrain Failure** | **0.92** | **0.85** |
| **Power Failure** | **0.65** | **0.30** |
| **Random Failures** | **0.0** | **0.0** |
| **Tool Wear Failure** | **0.82** | **0.0** |

The Leaky Naive Bayes model also demonstrates higher recall for most failure types, indicating that it captures more true positive instances.

|  |  |  |
| --- | --- | --- |
| **F1-score** | **Leaky Naive Bayes** | **Non-leaky Naive Bayes** |
| **Weighted avg** | **0. 99** | **0.95** |

TheF1-score is higher for the Leaky Naive Bayes model (Weighted avg: 0.99) compared to the Non-leaky Naive Bayes model (Weighted avg: 0.95).

Similarly, the Leaky Naive Bayes model outperforms the Non-leaky Naive Bayes model, which is expected due to the leakage.

**Model performance comparison with and without leakage (after applying SMOTE):**

**A screenshot of a computer screen

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Conclusion: -**

While the accuracy decreased across all four models, there were notable improvements in certain metrics for specific classes while others worsened. However, the models now exhibit a higher ability to discern random failures and tool wear failures with high precision and recall. It is evident that the previous metrics were misleading due to the low numbers of certain classes, resulting in inflated accuracy. The revised metrics now provide a more accurate representation of the models' performance.

**Explanation of data leakage impact on model performance:**

1. **Inflated Performance Metrics:** Since the leaked information is present in both the training and test datasets, the model achieves unrealistically high-performance during training and testing (especially in naïve bayes). However, this performance does not accurately reflect the model's ability to generalize to new, unseen data.
2. **Decreased Generalization:** When the model encounters new data without the leaked information, its performance drastically drops because it relies on patterns or correlations that are not present in the real-world scenario. This results in poor generalization and undermines the model's practical utility.
3. **Biased Predictions:** Data leakage introduces bias into the model, leading to biased predictions on unseen data. The model's decisions may be skewed towards the patterns present in the leaked information, potentially causing harm or misinformation in real-world applications.
4. **Lack of Robustness:** Models trained with data leakage lack robustness because they are overly sensitive to small changes in the input data. They may fail to adapt to new situations or environments, making them unreliable in real-world deployment.

**Top of Form**

**Description of data leakage detection techniques (bonus):**

1. **Permutation Importance:**
   * **Definition:** Permutation feature importance measures the increase in the prediction error of the model after we permuted the feature’s values, which breaks the relationship between the feature and the true outcome
   * **How it works:** measure the importance of a feature by calculating the increase in the model’s prediction error after permuting the feature. A feature is “important” if shuffling its values increases the model error, because in this case the model relied on the feature for the prediction. A feature is “unimportant” if shuffling its values leaves the model error unchanged, because in this case the model ignored the feature for the prediction.
   * **Limitations:** Permutation importance may not detect leakage if the feature interacts with other features in a non-linear or complex manner. It also requires careful consideration of feature interactions and model architecture.
2. **Shapley Values:**
   * **Definition :** The Shapley value provides a principled way to explain the predictions of nonlinear models common in the field of machine learning. By interpreting a model trained on a set of features as a value function on a coalition of players, Shapley values provide a natural way to compute which features contribute to a prediction  or contribute to the uncertainty of a prediction.
   * **How can it be used in leakage detection:** Shapley values assign credit to each feature for the model's prediction. Features that consistently receive high Shapley values across different samples may indicate leakage.
   * **Limitations:** Shapley values provide insights into feature importance but may not explicitly identify leakage. Interpretation of Shapley values requires understanding the model's behavior and the context of the data.
3. **Partial Dependence Plots:**
   * **Definition :** The partial dependence plot shows the marginal effect one or two features have on the predicted outcome of a machine learning model . A partial dependence plot can show whether the relationship between the target and a feature is linear, monotonic, or more complex. For example, when applied to a linear regression model, partial dependence plots always show a linear relationship.
   * **How it works:** Partial dependence plots show the relationship between a feature and the target while marginalizing over the values of other features. Sudden shifts or anomalies in these plots may indicate leakage.
   * **Limitations:** Partial dependence plots visualize the relationship between a single feature and the target but may not capture complex interactions or non-linear relationships.

**Ex:  
A graph of a number of years

Description automatically generated with medium confidence**

**References:**

* <https://docs.tibco.com/pub/sfire-dsc/6.5.0/doc/html/TIB_sfire-dsc_user-guide/GUID-07A78308-525A-406F-8221-9281F4E9D7CF.html#:~:text=Information%20Value%20analysis%20is%20a,of%20a%20specified%20dependent%20variable>.
* <https://en.wikipedia.org/wiki/Shapley_value#Formal_definition>
* <https://christophm.github.io/interpretable-ml-book/shapley.html#general-idea>
* <https://christophm.github.io/interpretable-ml-book/feature-importance.html>
* https://christophm.github.io/interpretable-ml-book/pdp.html