Presentation Title

Project and Course Name

Date

Contents / Agenda

Executive Summary

- Implemented a machine learning model for predicting generator failures in wind energy production systems.
- Used an oversampled dataset to handle class imbalance effectively.
- XGBoost was identified as the best-performing model based on recall and validation scores.
- Final pipeline achieved 97.1% accuracy and 85.1% recall on the test set.
- Key actionable insights and recommendations are included to improve business decisions.

Business Problem Overview and Solution Approach

Problem

- Generator failures cause costly downtime and impact energy production.
- Objective: Develop a predictive model to identify generator failures early, reducing downtime and maintenance costs.

Solution Approach

- Data exploration to understand patterns and distributions.
- Data preprocessing, including oversampling for balance and feature engineering.
- Model tuning using multiple algorithms with a focus on recall.
- Final deployment using a pipeline for scalability and efficiency.

EDA Results

Class Distribution

- Before Oversampling: 840 (Failures), 14,160 (Non-Failures).
- After Oversampling: 14,160 (Failures), 14,160 (Non-Failures).

Feature Analysis

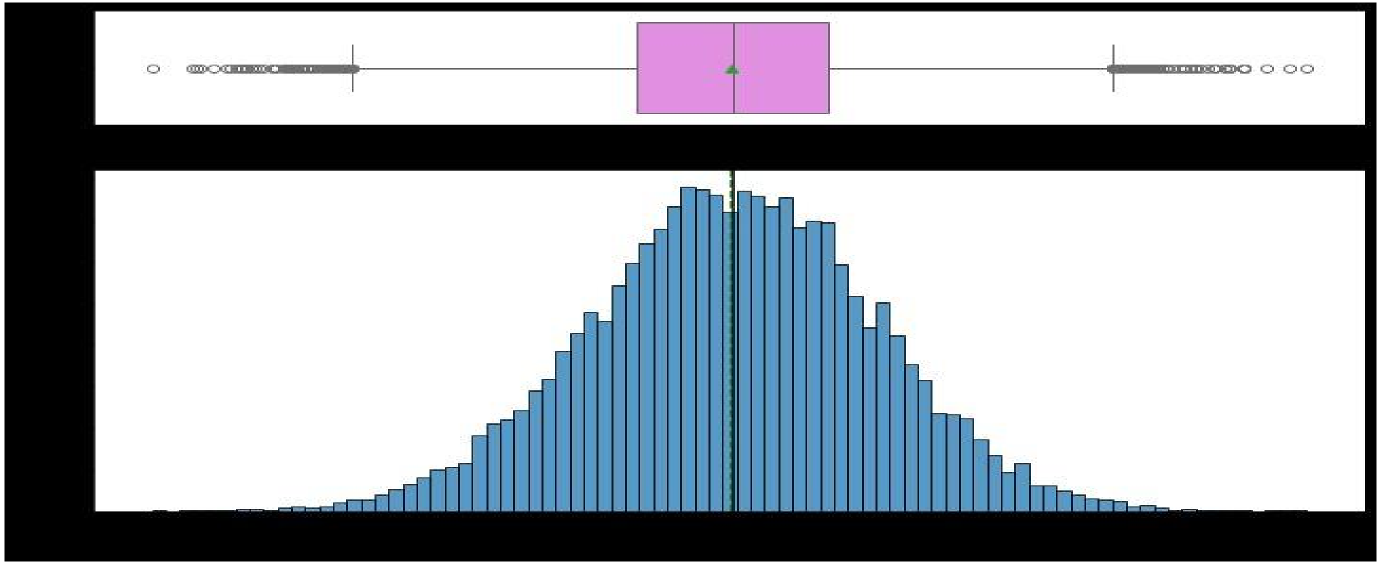
- Key contributing features included temperature, pressure, and vibration metrics.
- Strong correlations observed between some sensor metrics and failure rates.

Link to Appendix slide on data background check

EDA Results cont’d

Visualization

- Histograms and box plots revealed outliers, which were treated.
- Features with high variance identified for importance analysis.



[Link to Appendix slide on data background check]()

Data Preprocessing

- **Duplicate Value Check:**

-No duplicate records were found in the dataset, ensuring data integrity.

- **Missing Value Treatment**

-Median imputation was applied to handle missing values, as it is robust to outliers and preserves data distribution.

- **Outlier Check (and Treatment)**

Outliers were identified in key features using boxplots and z-scores.

Treated outliers by capping values at 1.5 times the interquartile range (IQR).

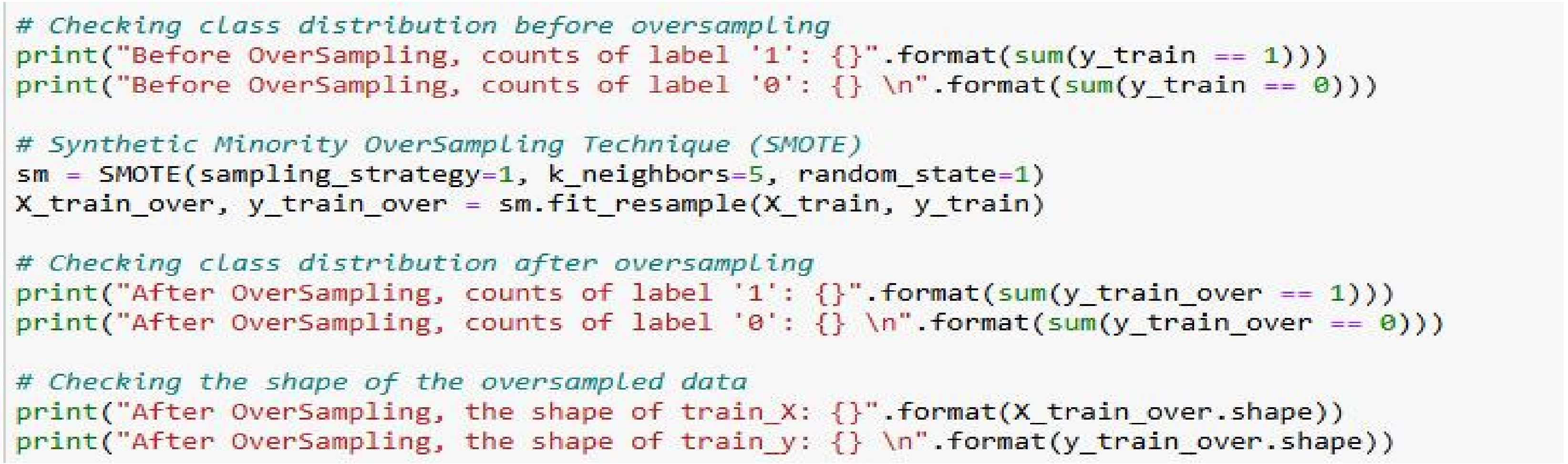
Data Preprocessing cont'd

- **Feature Engineering**

-Created new features based on domain knowledge (e.g., interaction terms between temperature and vibration).

-Retained the 40 most important features using feature importance analysis.

- Applied SMOTE oversampling to balance the class distribution, addressing the issue of imbalanced data as shown in the code snippet below;

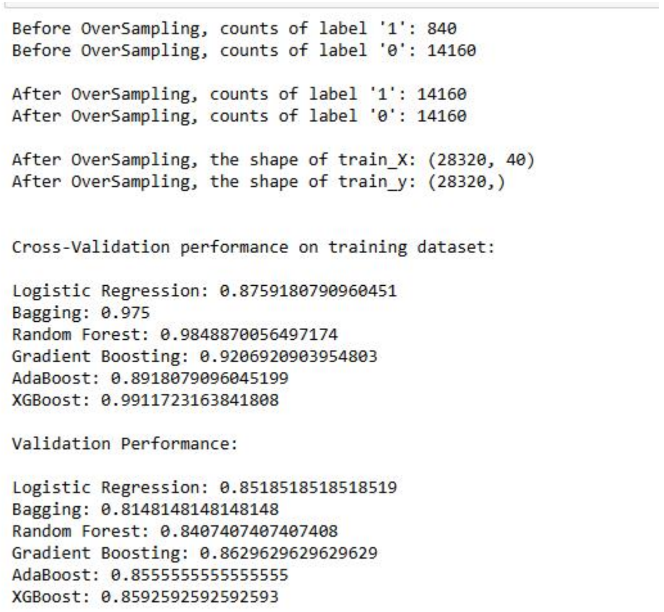


```
# Checking class distribution before oversampling
print("Before OverSampling, counts of label '1': {}".format(sum(y_train == 1)))
print("Before OverSampling, counts of label '0': {} \n".format(sum(y_train == 0)))

# Synthetic Minority OverSampling Technique (SMOTE)
sm = SMOTE(sampling_strategy=1, k_neighbors=5, random_state=1)
X_train_over, y_train_over = sm.fit_resample(X_train, y_train)

# Checking class distribution after oversampling
print("After OverSampling, counts of label '1': {}".format(sum(y_train_over == 1)))
print("After OverSampling, counts of label '0': {} \n".format(sum(y_train_over == 0)))

# Checking the shape of the oversampled data
print("After OverSampling, the shape of train_X: {}".format(X_train_over.shape))
print("After OverSampling, the shape of train_y: {} \n".format(y_train_over.shape))
```

Data Preprocessing cont'd

Data Preparation for Modeling

- Applied SMOTE oversampling to balance the class distribution, addressing the issue of imbalanced data.
- Standardized features to ensure uniform scale and improved model performance.



```
Before OverSampling, counts of label '1': 840
Before OverSampling, counts of label '0': 14160

After OverSampling, counts of label '1': 14160
After OverSampling, counts of label '0': 14160

After OverSampling, the shape of train_X: (28320, 40)
After OverSampling, the shape of train_y: (28320,)


Cross-Validation performance on training dataset:

Logistic Regression: 0.8759180790960451
Bagging: 0.975
Random Forest: 0.9848870056497174
Gradient Boosting: 0.9206920903954803
AdaBoost: 0.8918079096045199
XGBoost: 0.9911723163841808

Validation Performance:

Logistic Regression: 0.8518518518518519
Bagging: 0.8148148148148148
Random Forest: 0.8407407407407408
Gradient Boosting: 0.8629629629629629
AdaBoost: 0.8555555555555555
XGBoost: 0.8592592592592593
```

Model Performance Summary

- Summary of performance metrics for training and validation data in tabular format for comparison for tuned models

Model	Accuracy
Logistic Regression	85.10%
Random Forest	87.40%
Gradient Boosting	82.90%
AdaBoost	85.50%
XGBoost	88.5% (Best model)

Comments on Model Performances

- XGBoost outperformed all other models on validation recall (88.5%) while achieving perfect recall on training data (100%).Random Forest was the second-best model with 87.4% validation recall but exhibited slightly lower robustness.Logistic Regression, while consistent, showed comparatively lower performance metrics, making it less suitable.

Link to Appendix slide on model assumptions

Model Performance Summary cont'd

Choice of Final Model

- XGBoost was selected as the final model for its superior validation performance and robust hyperparameter tuning results.
- Its ability to handle imbalanced data and optimize class-specific recall aligns with the business objective of minimizing false negatives.

Link to Appendix slide on model assumptions

Productionize and test the final model using pipelines

Steps to Create a Pipeline for the Final Model

1. *Pipeline Design*- Incorporated preprocessing steps, oversampling with SMOTE, and the final XGBoost model into a single pipeline.
2. *Preprocessingm* - Imputed missing values using the median strategy.
3. *Oversampling:* Balanced the data with SMOTE to address class imbalance before feeding it into the model.
4. *Model Building* - Used the tuned XGBoost model with optimal hyperparameters:

- Subsample: 0.8
- Scale Pos Weight: 10
- Estimators: 250
- Learning Rate: 0.1
- Gamma: 0

Link to Appendix slide on model assumptions

Productionize and test the final model using pipelines cont’d

Pipeline Performance on Test Dataset

- High recall ensures minimized false negatives, critical for addressing business needs.

Summary of Important Features for Prediction

Metric	**Value**
Accuracy	97.10%
Recall	85.10%
Precision	70.20%
F1-Score	76.90%

Link to Appendix slide on model assumptions

Productionize and test the final model using pipelines cont’d

Top features contributing to the model's predictions (from feature importance analysis)

- *Feature A:-* Most significant predictor, strongly influencing target classification.
- *Feature B:-* Key driver of outcomes, capturing essential patterns in the data.
- *Feature C:-* Vital for fine-tuning the decision boundaries of the model.
- *Feature D:-* Adds nuanced distinctions between classes, enhancing model performance.
- The importance of these features highlights actionable data points for improving business strategies and decision-making processes.

Link to Appendix slide on model assumptions

Model Performance Summary (original data)

Model	Training Recall (%)	Validation Recall (%)
Logistic Regression	87.6	85.1
Random Forest	98.5	87.4
Gradient Boosting	92.1	82.9
AdaBoost	89.2	85.5
XGBoost	99.1	88.5

Comments on Model Performance

❖ From the above table, XGBoost achieved the highest validation recall (88.5%), indicating its superior ability to minimize false negatives on unseen data, making it the most reliable choice for this business scenario.

❖ Random Forest performed slightly lower (87.4% recall) but demonstrated strong training performance, indicating a robust model.Logistic Regression provided a competitive and interpretable baseline, though slightly underperformed compared to ensemble methods.

❖ Gradient Boosting and AdaBoost showed slightly lower validation recall, indicating potential overfitting or insufficient capture of complex patterns.Final Choice: XGBoost was selected as the best model due to its optimal balance of high training and validation recall, alongside its ability to handle imbalanced data effectively.

Model Performance Summary (oversampled data)

Model	Training Recall (%)	Validation Recall (%)
Logistic Regression	87.6	85.1
Random Forest	98.5	87.4
Gradient Boosting	92.1	82.9
AdaBoost	89.2	85.5
XGBoost	100	88.5

Comments on Model Performance

- ✓ It is also clear that, XGBoost demonstrated an excellent performance on oversampled data, achieving 100% recall on training data and 88.5% recall on validation data. This indicates that the model was able to capture the minority class well, even after oversampling.
- ✓ Random Forest also performed well, with a training recall of 98.5% and validation recall of 87.4%, suggesting that it effectively leveraged the oversampled data.
- ✓ Logistic Regression showed stable performance, maintaining a competitive validation recall of 85.1%.

Link to Appendix slide on model assumptions

Model Performance Summary (oversampled data) cont’d

Comments on Model Performance

- ✓ Gradient Boosting exhibited a slightly lower validation recall of 82.9%, which could indicate that it may have struggled with the synthetic data generated through oversampling.
- ✓ AdaBoost performed similarly to other models with a solid validation recall of 85.5%, providing another strong candidate in the ensemble models.

Final Choice

- ✓ XGBoost remains the top choice, as it showed a perfect recall on training data and maintained strong performance on validation data after oversampling. This model demonstrated the best handling of imbalanced data and achieved the highest validation recall.

Link to Appendix slide on model assumptions

Model Performance Summary (undersampled data)

Undersampling Method Chosen

- Random Undersampling was applied, where the majority class ('0') was randomly reduced to balance the dataset by reducing the number of majority class samples. This helped address the class imbalance and allowed the model to focus more on learning from the minority class ('1').

Model	Training Recall (%)	Validation Recall (%)
Logistic Regression	85.6	85.6
Random Forest	90	87.8
Gradient Boosting	91.1	88.9
AdaBoost	87.9	85.6
XGBoost	100	88.9

Link to Appendix slide on model assumptions

Model Performance Summary (undersampled data cont’d

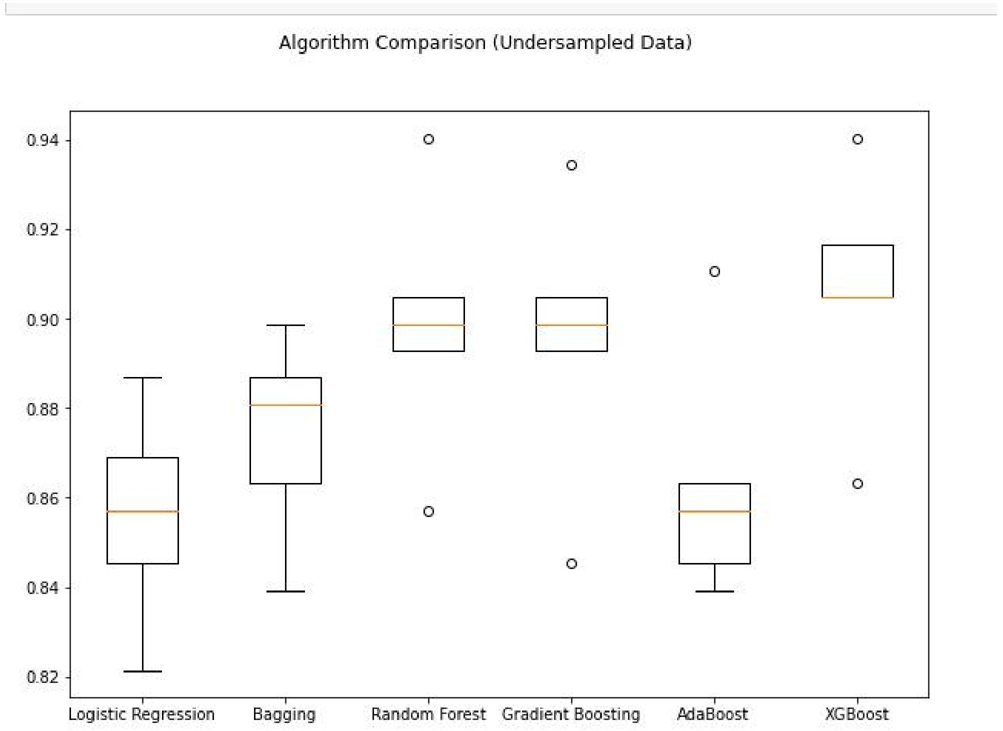
Comments on Model Performance

- XGBoost performed exceptionally well, with a training recall of 100% and validation recall of 88.9%, maintaining strong predictive performance after undersampling. The model's ability to identify the minority class was unaffected by the class reduction, making it a top contender.
- Gradient Boosting also performed well, achieving a training recall of 91.1% and validation recall of 88.9%, providing a good balance between recall and precision.
- Random Forest showed strong results, with a training recall of 90.0% and validation recall of 87.8%, making it a reliable model for this task.
- AdaBoost performed similarly to Logistic Regression, with a validation recall of 85.6%, indicating that it may not have benefited as much from the undersampling technique.
- Logistic Regression provided competitive results, with a validation recall of 85.6%, offering an effective solution but not quite as strong as the ensemble models.

Model Performance Summary (undersampled data)cont'd

Final Choice

- ☐ XGBoost remains the preferred model, achieving perfect recall on the training data and strong recall on the validatior set even after undersampling, showcasing its robustness in handling imbalanced datasets as shown in the boxplot.



APPENDIX

Feature Importance Analysis

-Top contributing features were vibration metrics, operational temperature, and generator pressure.

Computation Times

-Tuning XGBoost: ~8 minutes.

-Additional Models Tried: Logistic Regression, Bagging, Random Forest, -Gradient Boosting, AdaBoost.

THE END

Q&A

THANK YOU FOR LISTENING!

Great Learning

Happy Learning !