Microsoft Elevate

CAPSTONE PROJECT

PREDICTIVE MAINTENANCE MODEL

PRESENTED BY

STUDENT NAME: TEJASVI

COLLEGE NAME: SANSKARAM UNIVERSITY

DEPARTMENT: CSE

EMAIL ID: tejasvi12110@gmail.com

OUTLINE:

- Problem Statement
- Proposed System/Solution
- System Development Approach
- Algorithm & Deployment
- Result
- Conclusion
- Future Scope
- References

PROBLEM STATEMENT:

Industrial machinery downtime can lead to significant production losses and high maintenance costs. Traditional maintenance schedules are often reactive or time-based, which may not accurately prevent failures. This project aims to develop a machine learning model to predict equipment failure types in advance, enabling proactive maintenance and minimizing unexpected downtime.

PROPOSED SOLUTION:

- **Data Collection:** Gathered historical and real-time operational data from industrial machines, including parameters like air and process temperature, rotational speed, torque, tool wear, and machine type.
- **Data Preprocessing:** Cleaned the data, handled missing values, encoded categorical features (machine type) using one-hot encoding, and split the dataset into training and test sets.
- **Handling Imbalance:** Applied **SMOTE** to balance minority failure types, ensuring the model learns to predict all types of failures effectively.
- **Feature Scaling & Selection:** Standardized numerical features for uniformity and analyzed feature importance to identify key factors affecting machine failures.
- **Machine Learning Algorithms:** Trained a **Random Forest Classifier** due to its robustness, ability to handle categorical and numerical features, and interpretability.
- **Model Evaluation:** Assessed model performance using **accuracy, confusion matrix, classification report**, and feature importance charts to validate predictions.
- **Deployment & User Input:** Built an interface to receive real-time machine parameters and predict failure type, enabling proactive maintenance decisions.
- **Business Impact:** The solution reduces unexpected downtime, lowers maintenance costs, and improves production efficiency by predicting failures before they occur.

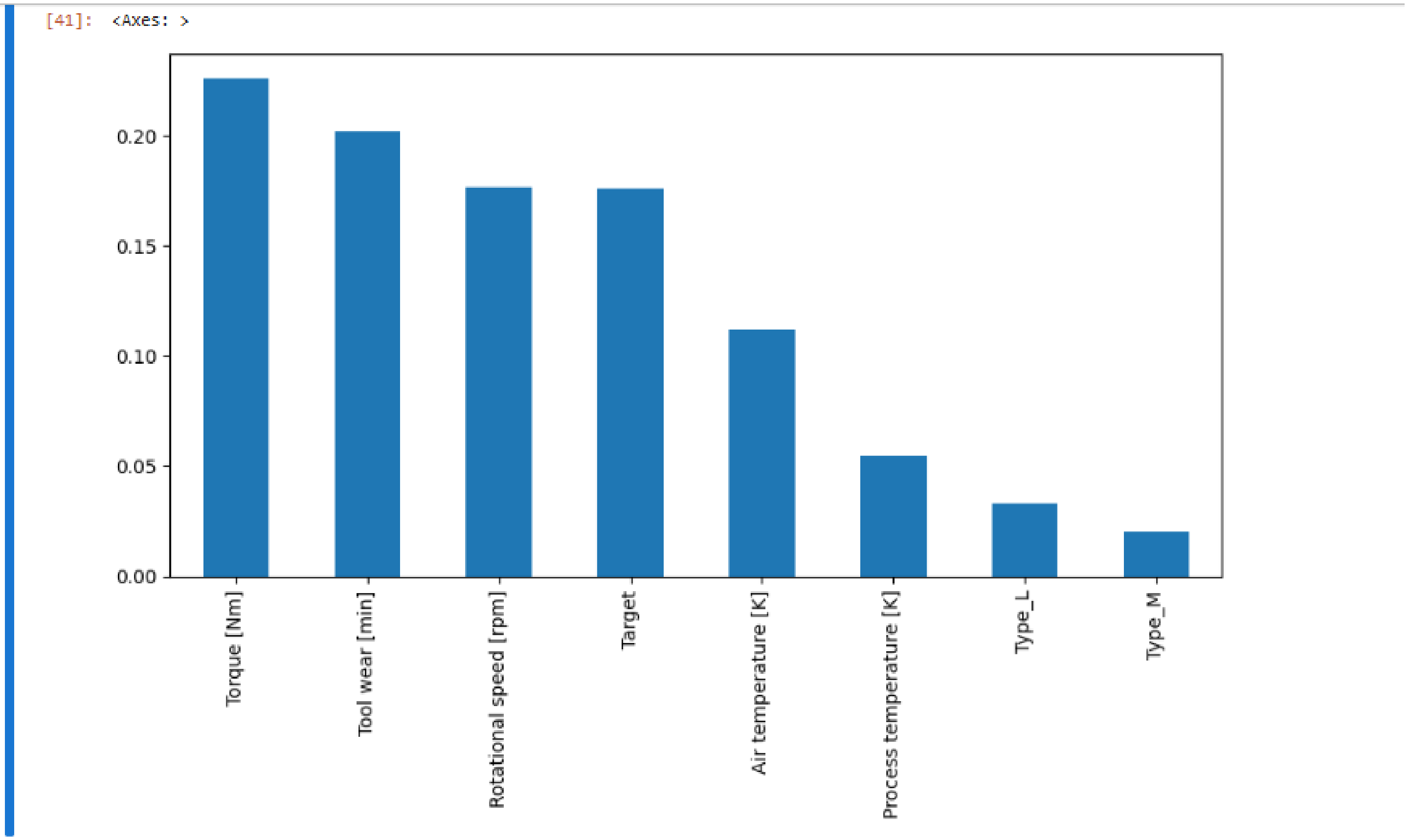
SYSTEM APPROACH:

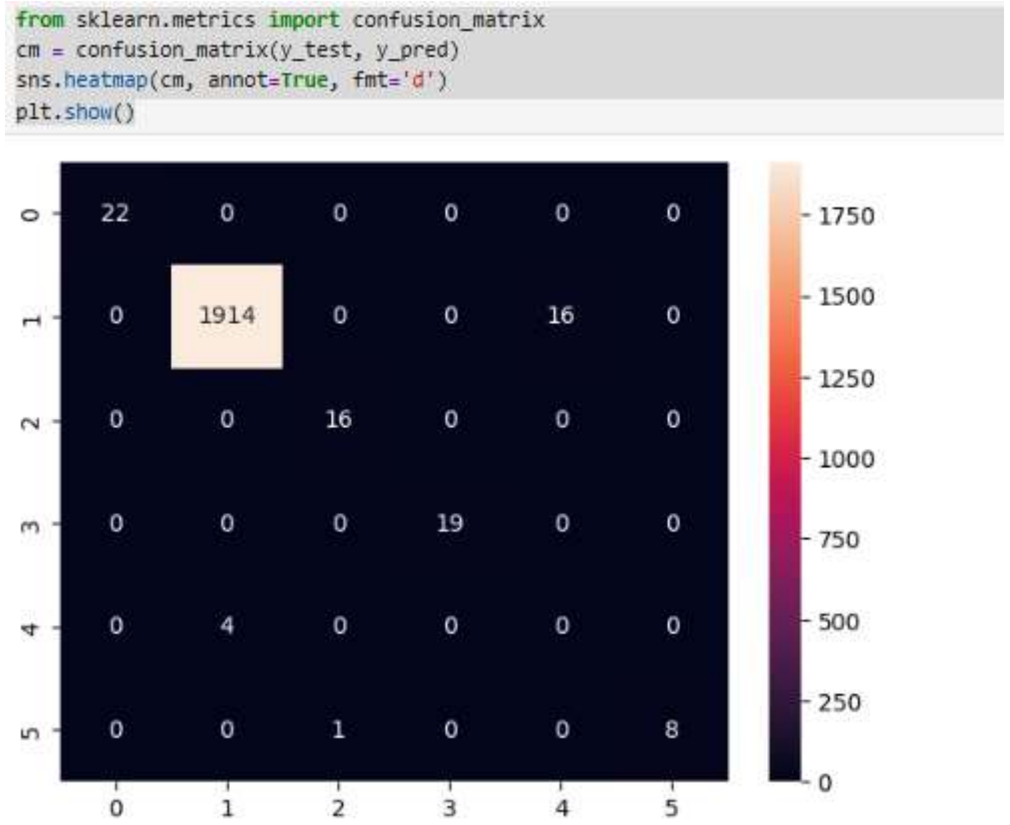
- **System Requirements:**
  - **Hardware:** Minimum 8 GB RAM, 2.5 GHz processor, and 10 GB free storage.
  - **Software:** Python 3.8+, Jupyter Notebook or any Python IDE, compatible OS (Windows/Linux/Mac).
- **Libraries & Tools Used:**
  - **Data Handling:** pandas, numpy
  - **Data Visualization:** matplotlib, seaborn
  - **Machine Learning:** scikit-learn (RandomForestClassifier, train_test_split, metrics)
  - **Imbalance Handling:** imblearn (SMOTE)
  - **Model Persistence:** joblib
  - **Hyperparameter Tuning:** GridSearchCV from scikit-learn
  - **Others:** Standard Python libraries for user input, file handling, and data preprocessing
- **System Workflow:**
  - Data Collection → 2. Preprocessing & Encoding → 3. Train-Test Split → 4. Handle Imbalanced Data → 5. Model Training → 6. Model Evaluation → 7. Deployment & User Prediction.

ALGORITHM & DEPLOYMENT:

- **Algorithm Selection:**
  - Chose **Random Forest Classifier** for its robustness, ability to handle both numerical and categorical features, high accuracy, and interpretability in predicting machine failure types.
- **Data Input:**
  - Collected operational parameters: Air & Process Temperature, Rotational Speed, Torque, Tool Wear, and Machine Type.
  - Categorical features (Machine Type) encoded using **one-hot encoding** for model compatibility.
- **Training Process:**
  - Split dataset into **training and test sets.**
  - Addressed class imbalance using **SMOTE** to ensure minority failure types are learned.
  - Trained Random Forest with hyperparameter tuning for optimal performance.
  - Evaluated model using **accuracy, classification report**, and feature importance.
- **Prediction Process / Deployment:**
  - Built an interface to accept **real-time machine input**.
  - Preprocessed user input (encoding & reindexing to match training features).
  - Model predicts **failure type** instantly, enabling proactive maintenance and reducing downtime.

RESULT:

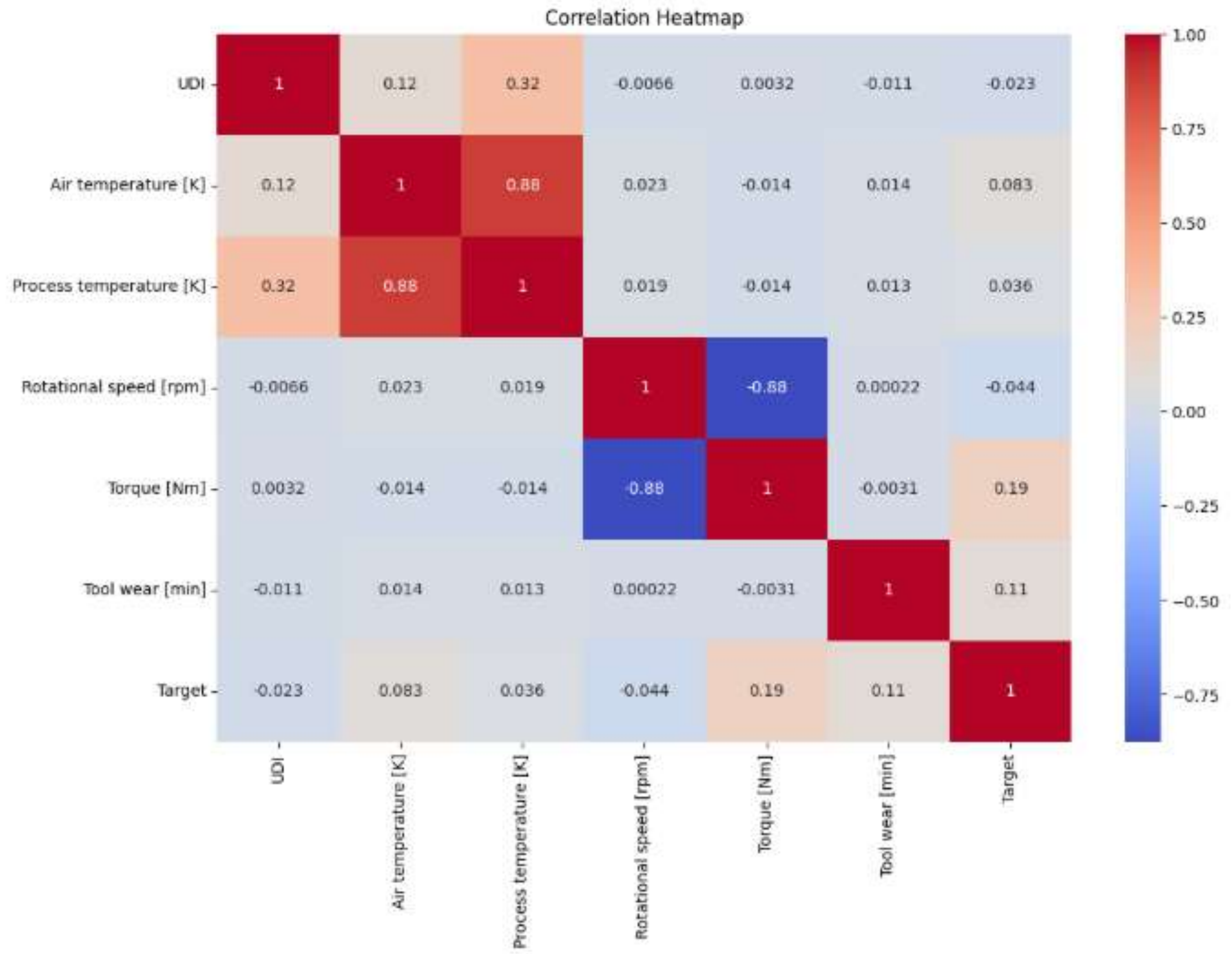

```
[41]:  <Axes: >
```


```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.show()
```

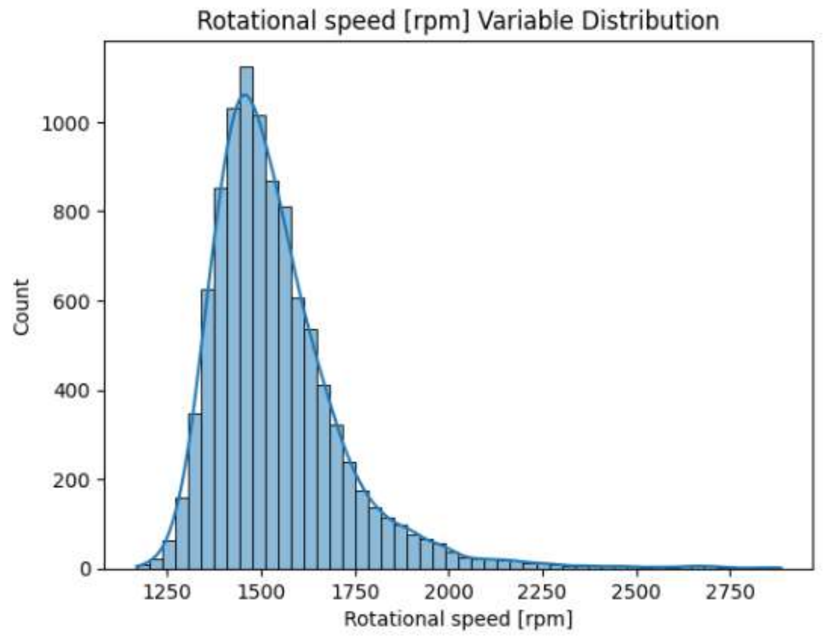
	0	1	2	3	4	5
0	22	0	0	0	0	0
1	0	1914	0	0	16	0
2	0	0	16	0	0	0
3	0	0	0	19	0	0
4	0	4	0	0	0	0
5	0	0	1	0	0	8

RESULT:

```
plt.show()
```



```
[12]: sns.histplot(df['Rotational speed [rpm]'], bins=50, kde=True)
      plt.title('Rotational speed [rpm] Variable Distribution')
      plt.show()
```



```
user_input_df = get_user_input()
Failure_Type = predict_maintenance(user_input_df)
print("Failure Type:", Failure_Type)
```

```
Enter machine operating details:

Air temperature (K):  298.1
Process temperature (K):  308.6
Rotational speed (rpm):  1551
Torque (Nm):  42.8
Tool wear (minutes):  0
Machine Type (L / M / H):  M
Failure Type: No Failure
```

CONCLUSION:

The predictive maintenance model successfully predicts the type of machinery failures using operational parameters, enabling proactive maintenance. By handling class imbalance and selecting important features, the Random Forest Classifier achieves high accuracy and reliable performance. Implementing this solution reduces unexpected downtime, minimizes maintenance costs, and improves overall production efficiency. This project demonstrates the practical benefits of machine learning in industrial maintenance decision-making.

FUTURE SCOPE:

The predictive maintenance system can be enhanced by integrating real time IoT sensor data for continuous monitoring. Advanced algorithms like XGBoost or deep learning models could further improve prediction accuracy. Predictive insights can be extended to schedule optimal maintenance and reduce energy consumption. Additionally, the system can be scaled for multiple machines and factories, supporting enterprise-wide smart maintenance strategies.

REFERENCES:

- Breiman, L. (2001). *Random Forests.* Machine Learning, 45(1), 5–32.
- Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). *SMOTE: Synthetic Minority Over-sampling Technique.* Journal of Artificial Intelligence Research, 16, 321–357.

GitHub Link: https://github.com/tejasvi1211/Predictive-Maintenance-Model

Microsoft Elevate

Thank You