Machine Learning and Logistic Regression in Failure Detection Problems in Vehicles

Portia Hungwe(22301924)
Masters Student, Applied AI for Digital Production
Deggendorf University of Technology, Cham, Germany
porthungwe@gmail.com

Nandakrishnan R(22309364)
Masters Student, Applied AI for Digital Production
Deggendorf University of Technology, Cham, Germany
rnandakrishnan2001@gmail.com

Dr. Sunil Survaiiya
Professor, Dept. of Applied Science
Deggendorf University of Technology, Cham, Germany
sunil.survaiya@th.deg.de

***Abstract*—Investigate the use of logistic regression in manufacturing failure detection, employing data from the APS failure and operational information from heavy trucks(scania). The study explores machine learning, linear, and Bayesian models, emphasizing the effectiveness of the XGBoost tree-based classifier for high-scoring classification. Logistic regression allows for a detailed analysis of influencing factors, while the Bayesian approach provides statistical distributions for model parameters, enabling probabilistic assessments such as risk analysis. The integration of these methodologies offers a comprehensive approach to failure detection. *Index Terms*—logistic regression; XGBoost; Bayesian inference; failure detection.**

I. Introduction

Checking intently the manufacturing processes is significant, as they progress through the assembling processes. Scania records information at each step along its assembly lines. They can apply progressed examination to move along these assembling processes. Nonetheless, the complexities of the information and intricacies of the creation line present a few issues for current strategies. The assignment of inner disappointment expectation is a strategic relapse issue. Groupings are assessed utilizing the Matthews connection coefficient (MCC) between the observed and the predicted data. The MCC is given where TN is the number of true negatives, and FN is the number of false negatives. True Positives (TP) represent instances correctly identified as positive, while False Positives (FP) indicate instances incorrectly classified as positive. and the information set contains an enormous number of anonymized highlights. This study aims to consider unique approaches for time-producing portions of failure modeling. AI calculations make it conceivable to track down designs in the information. This approach can give high accuracy in logistic regression. We can characterize the significance of each element however it can't give us any data about the impact of each variable.

$$MCC = \frac{(TP \cdot TN) - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

Fig. 1. MCC equation

II. Dataset

The dataset comprises of information gathered from Sca- nia trucks(heavy) in regular use. The framework is focused on the Air Pressure system (APS) which creates compressed air that is used in different capabilities in a truck, such as slowing down and gear changes. The datasets' positive class comprises of part for a particular part of the APS framework. The negative class comprises trucks with failures for parts not connected with the APS. The information comprises a subset of every accessible data, chosen by specialists.

A. Data cleaning

Data cleaning is a critical step that includes recognizing errors also, correcting disparities and irregularities inside datasets to guarantee precise and normalized data for viable integration. Python gives a flexible arrangement of libraries and instruments for data cleaning assignments. This incorporates utilizing pandas for information profiling, taking care of missing qualities, and tending to copies. Exceptions can be identified and treated utilizing factual strategies, and the sci-pack learn library offers capabilities for standardization and normalization. Also, pandas can be utilized to determine irregularities in data. Furthermore, pandas can be utilized to determine irregularities in units and data formats, while characterizing and approving data quality measurements can be accomplished utilizing custom Python code or other specific libraries. The iterative idea of data cleaning in Python considers constant refinement of the

cycle, guaranteeing that coordinated datasets fulfill great guidelines and work with significant examinations in linear applications.

B. Data transformation

Python code

```
import pandas as pd
from sklearn.preprocessing import LabelEncoder
df= pd.read_csv("your_dataset.csv")
df.fillna(df.mean(), inplace=True)
mapping_dict = {"string1": 1, "string2": 2}
df["categorical_column"] =
df["categorical_column"].replace(mapping_dict)
df=pd.get_dummies(df,columns=["categorical_column"])
le=LabelEncoder()
df["ordinal_column"] =
le.fit_transform(df["ordinal_column"])
```

C. Data categorizing

Here, taking 60000 models altogether 59000 have a place with the negative class, and 1000 positive class. The test set contains 16000 models. But here in our project, we changed the information focus to preparing set being 40000 altogether, with 39000 being negative and 1000 positive class. The test set contained 20/of the preparation set. The quantity of Traits was 171 also, their names were been anonymized for restrictive reasons. This program - operational data and APS failure for Scania Trucks is free programming and it allows rearrangement and additional changes under the details of the The GNU Overall populace Permit is disbursed via way of means of the Free Programming Establishment, either variant 3 of the Permit, or on the other hand (at your choice) any later rendition.

III. Methodology

This process was done in Python and all the codes are provided. We did not have to split the data as it was already done by the data source provider. Model creation. For the industry failure prediction problems, several models first come into play, and from those, we picked a few. The goal was to build two different models using different approaches and then make a conclusion as to which model fit the provided data the best.

The advantage of a Bayesian way to deal with failure and inconsistency examinations is that it is many times the situation is a failure and inconsistency examinations that data is either restricted or so far-reaching that it is challenging to carry concentration to a key root cause. Consequently, a restrained methodology consolidating the underlying driver trees (Ishikawa Graphs) is generally taken to create and follow the underlying driver speculations and examinations. During the examination, statistical instruments can be utilized to assess different speculations of failure.

Here, the root cause is achieved sometimes only after costly and broad efforts to the number of main driver speculations. Also, there is data failure(limited), and setting up accelerated life is much of the time vital tests including many examples to instigate failure under controlled conditions so that a measurably critical number of failures can be gotten. Underlying driver examinations are shortened to and quote; most likely cause" in view of the proof accessible and well-qualified assessment. Bayesian analysis permits test or perception information to be joined with earlier data to create a back gauge of probability. It very well may be a device that gives a few advantages to the main driver assurance process. The main advantage is to give a gauge of the probability that specific speculations are valid based on the restricted information accessible. This can give valuable course to the analysis failure.

IV. Instructions

	Positive	Negative
Positive		Cost_1
Negative	Cost_2	

TABLE I: Cost-metric of miss-classification

The cost associated with a prediction model comprises two components: "Cost 1," representing the expense of performing an unnecessary check by a mechanic at a workshop, multiplied by the number of instances with type 1 failure, and "Cost 2," denoting the cost of missing a faulty truck leading to a breakdown, multiplied by the number of instances with type 2 failure.

In this context, "Cost_1" represents the expense incurred when a mechanic conducts an unnecessary check at a workshop. On the other hand, "Cost_2" pertains to the financial impact associated with overlooking a defect in a truck, which could potentially lead to a breakdown.

Total_cost = Cost_1*No_Instances+Cost_2*No_Instances.

The primary objective of this study is to draw accurate inferences that aid in the identification of incorrect error types, ultimately assisting Scania Trucks in preventing unnecessary costs.

V. Logistic regression model

This model was built and trained in the training dataset, after that it was loaded into a different Jupiter notebook where the testing dataset was uploaded so that the model could be tested and validated on that specific dataset.

```
Confusion Matrix:
[[7795    0]
[   0  203]]
```

Fig. 2. Confusion matrix-LRM

According to this confusion matrix, there were 7795 true positives and 203 true negatives. There is no sign of false alarms in the data i.e. zero false positives and negatives.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.99      7795
           1       0.00      0.00      0.00       203

    accuracy                           0.97      7998
   macro avg       0.49      0.50      0.49      7998
weighted avg       0.95      0.97      0.96      7998
```

Fig. 3. Classification report-LRM

Accuracy: 97.46%
This model accuracy means that the model fits well to the test data.

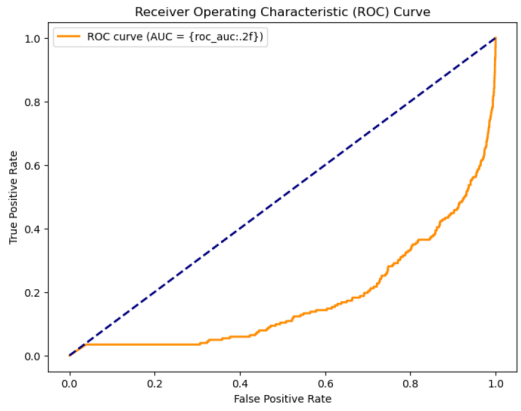

Fig. 4. ROC curve-LRM

Even though the model accuracy as well as the mean CV score are quite good, this graph says otherwise. However, the model is still a good fitting model according to the mean score that is quite close to 1. For this reason, we have to move from try using machine learning algorithms to analyze this data.

VI. • XGBoost(Extreme Gradient Boosting) Classifier

In the same way as the Logistic regression model, this model was built and trained in the training dataset, after that it was loaded into a different Jupiter notebook where the testing dataset was uploaded so that the model could be tested and validated on that specific dataset.

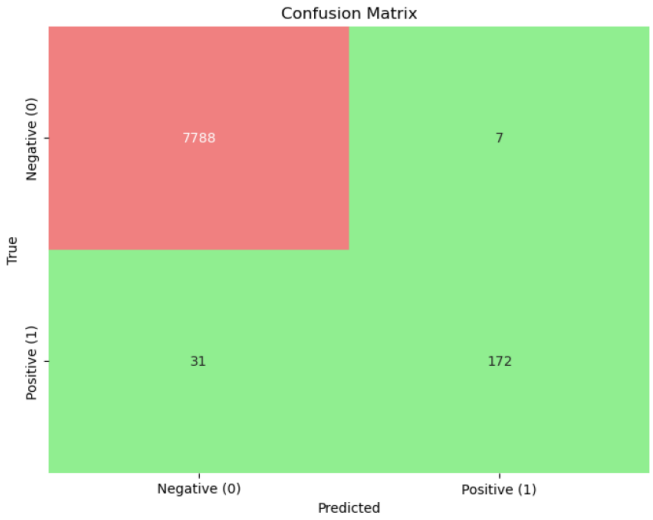

Fig. 5. Confusion matrix-XGB

Now this matrix includes some false alarms that were not detected by the confusion matrix of the other model last used.

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7795
           1       0.96      0.85      0.90       203

    accuracy                           1.00      7998
   macro avg       0.98      0.92      0.95      7998
weighted avg       1.00      1.00      1.00      7998
```

Fig. 6. Classification report-XGB

According to this confusion matrix

Accuracy: 99.52%

This model accuracy percentage suggests a strong performance of the XGBoost model. The model generalizes well to unseen data, it appears to be effective and reliable.

This ROC (Receiver Operating Characteristic) curve does represent how a typical good ROC curve should

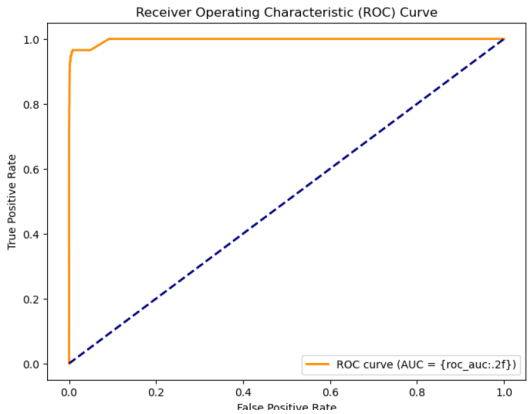

Fig. 7. ROC curve-XGB

look like, i.e. closer to the top left corner since a higher AUC is generally desirable. This gives insights into the model's ability to discriminate between the positive and the negative classes across different thresholds. Even though this model proves strong performance on the data, it is still required to cross-validate it.

Mean Accuracy: 97.99%

A. Bayesianmodeling

This model is involved in the XGBoost model processing during model cross-validation. By optimizing using the python sklearn.model_selectionmodule in python, there is the utilization of the Bayesian Optimization.

B. Flow Chart

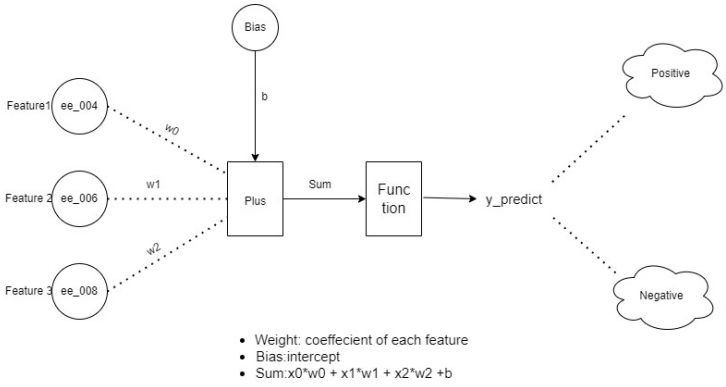

Fig. 8. Flow chart of key device

VII. User Interface

Note: To make some logistic regression analysis predictions using our model, one can scan the QR code below after running the code on any local machine(local host).

Fig. 9. link:https://127.0.0.1:5000

VIII. Results

The second model (XGBoost classifier) demonstrates an excellent performance for class 0, as indicated by the precision, recall, and F1 score percentages which are all 1. This suggests that the model indicates and classifies all instances of class 0. For class 1, while the precision is 0.96 and the recall of 0.85 is not perfect, the F1 score of 0.90 balances the precision and recall values. The macro average, which calculates the unweighted mean of precision, recall, and F1 score across both classes, is high at 0.98, indicating strong overall model performance. In summary, the XGBoost model excels in classifying class 0 instances and performs well for class 1, achieving a high macro average across both classes. The best model for this failure detection problem is the XGBoost model since it has been able to discriminate between the positive and negative classes across all thresholds. Even though the Logistic Regression model was overall a well-performing model it still had shortcomings that the other machine learning model was able to address and this is what would help Scania Trucks solve the problem and avoid losing money for unnecessary repairs, as well as detecting which trucks exactly have an APS problem and which ones do not.

IX. References

References

[1] Tianqi Chen and Carlos Guestrin. (2016) 'XGBoost: A Scalable Tree Boosting System.' arXiv:1603.02754v3 [cs.LG] 10 Jun 2016.

[2] J. Friedman (2001). "Greedy function approximation: a gradient boosting machine.", Annals of Statistics, 29(5):1189–1232.

[3] J. Friedman (2002). "Stochastic gradient boosting.", Computational Statistics & Data Analysis, 38(4):367–378.

[4] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. (2008). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, Vol. 33(1), 1-22 Feb 2010.

[5] Ziparo, V.A. Iocchi, L. (2006). Petri net plans. Proc. of Fourth International Workshop on Modelling of Objects, Components, and Agents (MOCA), Turku, Finland, pp. 267–290, Bericht 272, FBI-HH-B272/06.

[6] Martyn Plummer. (n.d.). JAGS Version 3.4.0 user manual. URL: http://sourceforge.net/projects/mcmcjags/files/Manuals/3.x/jags_user_manual.pdf

[7] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. Journal of Statistical Software, Vol. 39(5), 1-13.
[8] Kruschke, John. (2014). Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan. Academic Press.
[9] Stanford Statistics Technical Report. (n.d.). Glmnet Vignette. URL: http://www.stanford.edu hastie/glmnet/glmnet_alpha.html
[10] Robert Tibshirani, Jacob Bien, Jerome Friedman, Trevor Hastie, Noah Simon, Jonathan Taylor, Ryan J. Tibshirani. (2010). Strong Rules for Discarding Predictors in Lasso-type Problems. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 74(2), 245-266.
[11] Machine learning, linear and Bayesian models for logistic regression in failure detection problems — IEEE Conference Publication — IEEE Xplore.

X. AUTHOR CONTRIBUTIONS

1. Nandakrishnan Rajeev - Data collection, testing, report writing
2. Portia Hungwe - Data integration, data analysis, training Research paper collection, presentation, and GUI are collective efforts.

XI. APPENDIX: SOFTWARE CODE

Machine Learning and Logistic Regression in Failure Detection Problems in vehicles have two codes.

A. Linear regression train model

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_curve
df = pd.read]_csv('aps_failure_training_set_new.csv')
df.head(5)
X = df[['ee_004', 'ee_006', 'ee_008']]
y = df['class']
model = LogisticRegression()
model.fit(X, y)
import joblib
joblib.dump(model, 'logistic_regression_model.pkl')
filename = 'logistic_regression model.pkl'
joblib.dump(model, filename)
```

B. Logistic regression analysis

```
pd.set_option('display.max_columns', None)
print(f'The data set have df.shape[0] rows and df.shape[1] columns')
print("**" * 30)
print(" " * 17, "Data set Information")
print("**" * 30)
print(df.info())
df.isnull()
df.isnull().any()
X_test = df[['ee_004', 'ee_006', 'ee_008']]
y_test = df['class']
import joblib
import os
loaded_model=joblib.load('logistic_regression model.pkl')
X_test = df[['ee_004', 'ee_006', 'ee_008']]
y = df['class']
loaded_model = LogisticRegression()
loaded_model.fit(X_test, y)
y_pred = loaded_model.predict(X_test)
print("Predictions:", y_pred)
y_pred = loaded_model.predict(X_test)
y_prob = loaded_model.predict_proba(X_test)[:, 1]
conf_matrix = confusion_matrix(y, y)
print("Confusion Matrix:
n", conf_matrix)
class_report = classification_report(y, y_pred)
print("Classification Report:
n", class_report)
accuracy = accuracy_score(y, y_pred)
print(f'accuracy: accuracy * 100:.2f
fpr, tpr, thresholds = roc_curve(y, y_prob)
roc_auc = roc_auc_score(y, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color=f'darkorange',lw=2, label='ROCcurve(AUC=roc_auc:.2f)')
plt.plot([0, 1], [0, 1], color='navy',lw=2,linestyle='–')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
from sklearn.model_selection import cross_val_score
cross_val_scores=cross_val_score(loaded_model, X_test, y, cv=5)
print("Cross-validation scores:", cross_val_scores)
print("Mean CV score:", cross_val_scores.mean())
```

C. XG boost train model

```
pip install xgboost pandas sci-kit-learn
import pandas as pd
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix, classification_report, roc_auc_score, roc_curve
df = pd.read_csv('aps_failure_training_set_new.csv')
df.head(5)
X = df[['ee_004', 'ee_006', 'ee_008']]
y = df['class']
model = XGBClassifier()
model = XGBClassifier(objective='binary:logistic')
model.fit(x, y)
import joblib
```

```
joblib.dump(model, 'xgboost_model.pkl')
filename = 'xgboost_model.pkl'
joblib.dump(model, filename)
```

D. XG boost test analysis

```
pd.set_option('display.max_columns', None)
print(f'The data set have test_data.shape[0] rows and test_data.shape[1] columns')
print("**" * 30)
print(" " * 17, "Data set Information")
print("**" * 30)
print(test_data.info())
test_data.isnull()
test_data.isnull().any()
_test=test_data[['ee_004', 'ee_006', 'ee_008']]
y = test_data['class']
import joblib
import os
loaded_model=joblib.load('xgboost_model.pkl')
loaded_model=xgb.Booster()
loaded_model=XGBClassifier(objective='binary:logistic')
loaded_model.fit(X_test, y)
y_pred =loaded_model.predict(X_test)
accuracy = accuracy_score(y, y_pred)
print(f'Accuracy: accuracy * 100:.2f
class_report = classification_report(y, y_pred)
print("Classification Report:
n", class_report)
cm = confusion_matrix(y, y_pred)
import matplotlib. colors as mcolors
cmap_custom = mcolors.ListedColormap(['light green, 'light green,' light coral, 'light coral'])
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True,fmt='d',cmap=cmap_custom,
cbar=False,
xticklabels=['Negative (0)', 'Positive (1)'],
yticklabels=['Negative (0)', 'Positive (1)'])
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
y_pred = loaded_model.predict(X_test)
y_prob = loaded_model.predict_proba(X_test)[:, 1]
from sklearn import metrics
y_true = [0, 1, 0, 1, 1, 0]
y_score = [0.2, 0.8, 0.3, 0.7, 0.9, 0.1]
roc_auc = metrics.roc_auc_score(y_true, y_score)
fpr, tpr, thresholds = roc_curve(y, y_prob)
roc_auc = roc_auc_score(y, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color=f'darkorange', lw=2, label='ROC curve (AUC = roc_auc:.2f)')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='–')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.show()
from sklearn.model_selection import StratifiedKFold, cross_val_score
cv=StratifiedKFold(n_splits=5,shuffle=True, random_state=42)
scores=cross_val_score(loaded_model,X_test,y,scoring='accuracy', cv=cv)
cross_val_scores=cross_val_score(loaded_model, X_test, y, cv=5)
print("Cross-validation scores:", cross_val_scores)
print("Mean Accuracy: %.2f%%" % (scores.mean() *100))
```

E. GUI code

```
from flask import Flask, render_template, request
import pandas as pd
import pickle
import joblib
app = Flask(__name__)
# Loading the Pre-trained model using Pickle
#model = joblib.load(r"C:\Users\PC\logistic \_regression model.pkl")
model = joblib.load(r"C:_model.pkl")
@app.route('/', methods=['GET', 'POST'])
def main():
prediction = None
if request.method == 'GET':
return(render_template('main.html'))
#Getting Form Input
if request.method == 'POST':
ee_004 = float(request.form['ee_004'])
ee_006 = float(request.form['ee_006'])
ee_008 = float(request.form['ee_008'])
print(ee_004)
Putting Form input in a data frame
input_variables = pd.DataFrame([[ee_004, ee_006, ee_008]],
columns=['ee_004', 'ee_006', 'ee_008'], dtype=float)
# Predicting the Wine Quality using the loaded model
prediction = model.predict([[ee_004, ee_006, ee_008]])[0]
prediction = 0
return render_template('main.html',
result=prediction )
@app.route('/model evaluation, methods=['GET'])
def model():
accuracy = 99.52
df_clas_report =
pd.read_csv(r'C:_outreport.csv')
print(df_clas_report.columns)
return
render_template('modelevaluation.html',accuracy=accuracy,
table_pred_cols=df_clas_report.
```

```
columns.values,
table_pred_data= [df_clas_report.to_html(classes='data',
index=False, justify='center')])
if __name__ == '__main__':
```