

Machine Failure Prediction Project

I have used a logistic regression model to analyse the relationship between sensor readings and the likelihood of machine failure.

The image displays two sequential screenshots of a Google Colab notebook titled 'Untitled1.ipynb'. The interface includes a file explorer on the left, a code editor in the center, and a system status bar at the bottom.

Top Screenshot: Shows the initial code execution. The first cell imports necessary libraries: pandas, numpy, matplotlib, seaborn, sklearn, and warnings. The second cell loads the dataset 'data (1).csv' and prints the number of missing values, which are zero for all features.

```
[1] import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, roc_auc_score
import warnings
warnings.filterwarnings('ignore')

[2] df = pd.read_csv("/content/data (1).csv", encoding='ascii')

print("Missing values:")
print(df.isnull().sum())
```

Missing values:
footfall 0
tempMode 0
AQ 0
USS 0
CS 0
VOC 0
RP 0
IP 0
Temperature 0
fail 0
dtype: int64

Bottom Screenshot: Shows the next steps in the notebook. The third cell prints the basic statistics of the dataset, and the fourth cell calculates the correlation matrix.

```
[4] print("\nBasic statistics:")
print(df.describe())

[5] corr_matrix = df.corr()
```

Basic statistics:

	footfall	tempMode	AQ	USS	CS
count	944.000000	944.000000	944.000000	944.000000	944.000000
mean	306.381356	3.727754	4.325212	2.939619	5.394068
std	1082.606745	2.677235	1.438436	1.383725	1.269349
min	0.000000	0.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	3.000000	2.000000	5.000000
50%	22.000000	3.000000	4.000000	3.000000	6.000000
75%	110.000000	7.000000	6.000000	4.000000	6.000000
max	7380.000000	7.000000	7.000000	7.000000	7.000000

	VOC	RP	IP	Temperature	fail
count	944.000000	944.000000	944.000000	944.000000	944.000000
mean	2.842161	47.043432	4.565678	16.331568	0.416314
std	2.273337	16.423130	1.599287	5.574781	0.493208
min	0.000000	19.000000	1.000000	1.000000	0.000000
25%	1.000000	34.000000	3.000000	14.000000	0.000000
50%	2.000000	44.000000	4.000000	17.000000	0.000000
75%	5.000000	58.000000	6.000000	21.000000	1.000000
max	6.000000	91.000000	7.000000	24.000000	1.000000

Untitled1.ipynb - Colab

colab.research.google.com/drive/1950erNAPwh0WAQH9b_ktpa7RZxY1WYtA#scrollTo=zJjCv6wVuoxv

Files

- sample_data
- confusion_matrix_plot.png
- correlation_heatmap.png
- data (1).csv
- failure_distribution.png
- feature_importance.png

```
[5] corr_matrix = df.corr()

[6] plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title('correlation Heatmap')
plt.tight_layout()
plt.savefig('correlation_heatmap.png')
plt.close()

[7] plt.figure(figsize=(8, 6))
sns.countplot(x='fail', data=df)
plt.title('Distribution of Machine Failures')
plt.savefig('failure_distribution.png')
plt.close()

print("\n
class distribution:")
print(df['fail'].value_counts(normalize=True))

Class distribution:
fail
0    0.583686
1    0.416314
Name: proportion, dtype: float64

[10] X = df.drop('fail', axis=1)
```

completed at 6:11 PM

Untitled1.ipynb - Colab

colab.research.google.com/drive/1950erNAPwh0WAQH9b_ktpa7RZxY1WYtA#scrollTo=zJjCv6wVuoxv

Files

- sample_data
- confusion_matrix_plot.png
- correlation_heatmap.png
- data (1).csv
- failure_distribution.png
- feature_importance.png

```
[10] X = df.drop('fail', axis=1)
y = df['fail']

[11] X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

[12] scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

model = LogisticRegression(random_state=42)
model.fit(X_train_scaled, y_train)

LogisticRegression
LogisticRegression(random_state=42)

[14] y_pred = model.predict(X_test_scaled)

[15] print("\n
Classification Report:")
print(classification_report(y_test, y_pred))

Classification Report:
precision    recall  f1-score   support

0.00         0.00         0.00         100
0.00         0.00         0.00         100
```

completed at 6:11 PM

Untitled1.ipynb - Colab

colab.research.google.com/drive/1950erNAPwh0WAQH9b_ktpa7RZxY1WYtA#scrollTo=zJjCv6wVuoxv

Files

- sample_data
- confusion_matrix_plot.png
- correlation_heatmap.png
- data (1).csv
- failure_distribution.png
- feature_importance.png

```
[15] print("\n
Classification Report:")
print(classification_report(y_test, y_pred))

Classification Report:
              precision    recall  f1-score   support

      0       0.90      0.85      0.87        102
      1       0.84      0.89      0.86         87

   accuracy      0.87      0.87      0.87        189
  macro avg       0.87      0.87      0.87        189
 weighted avg       0.87      0.87      0.87        189

[16] print("\n
Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))

Confusion Matrix:
[[87 15]
 [10 77]]

roc_auc = roc_auc_score(y_test, model.predict_proba(X_test_scaled)[: , 1])
print(f"\n
ROC AUC Score: {roc_auc:.4f}")
```

0s completed at 6:11 PM

Untitled1.ipynb - Colab

colab.research.google.com/drive/1950erNAPwh0WAQH9b_ktpa7RZxY1WYtA#scrollTo=zJjCv6wVuoxv

Files

- sample_data
- confusion_matrix_plot.png
- correlation_heatmap.png
- data (1).csv
- failure_distribution.png
- feature_importance.png

```
[18] feature_importance = pd.DataFrame({
    'feature': X.columns,
    'importance': abs(model.coef_[0])
})
feature_importance = feature_importance.sort_values('importance', ascending=False)

[21] plt.figure(figsize=(10, 6))
sns.barplot(x='importance', y='feature', data=feature_importance)
plt.title('Feature Importance')
plt.tight_layout()
plt.savefig('feature_importance.png')
plt.close()

[22] print("\n
Top 5 most important features:")
print(feature_importance.head())

Top 5 most important features:
   feature  importance
5      VDC      2.420079
3      USS      1.297031
2      AQ       0.825277
4      CS       0.603889
0  footfall      0.266227
```

0s completed at 6:11 PM

The screenshot shows a Google Colab notebook interface. The left sidebar displays a file explorer with a folder named 'sample_data' and several image files: 'confusion_matrix_plot.png', 'correlation_heatmap.png', 'data (1).csv', 'failure_distribution.png', and 'feature_importance.png'. The main code area contains three code cells. The first cell shows a table of feature importance values. The second cell imports the necessary libraries: pandas, matplotlib.pyplot, and seaborn. The third cell defines a function to calculate the confusion matrix and generate a heatmap plot, saving it as 'confusion_matrix_plot.png'.

feature	importance	
5	VOC	2.428879
3	USS	1.297031
2	AQ	0.825277
4	CS	0.603889
0	footfall	0.266227

```
[22] feature importance
5     VOC      2.428879
3     USS      1.297031
2     AQ       0.825277
4     CS       0.603889
0  footfall    0.266227

[23] import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

[25] import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

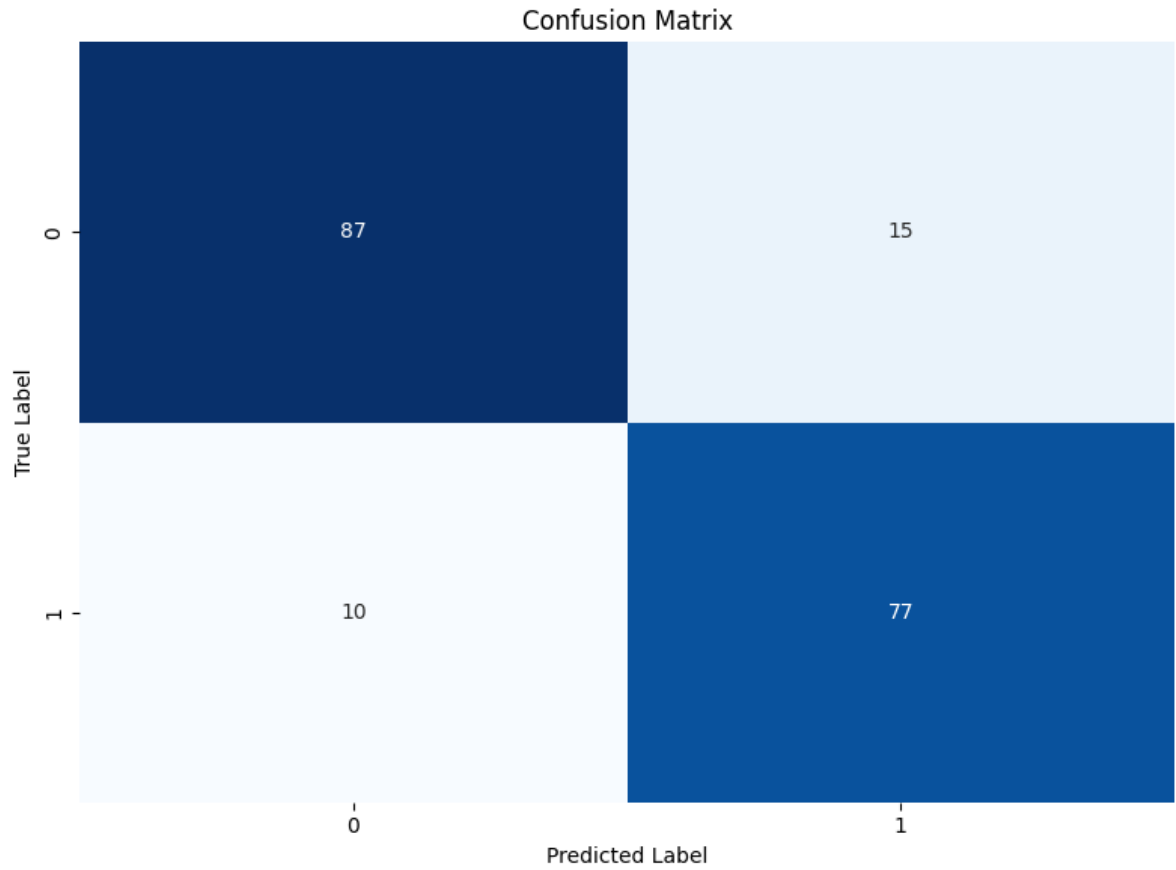
cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.tight_layout()
plt.savefig('confusion_matrix_plot.png')
plt.show()
```

The dataset has no missing values.

Here are the results and visualizations:

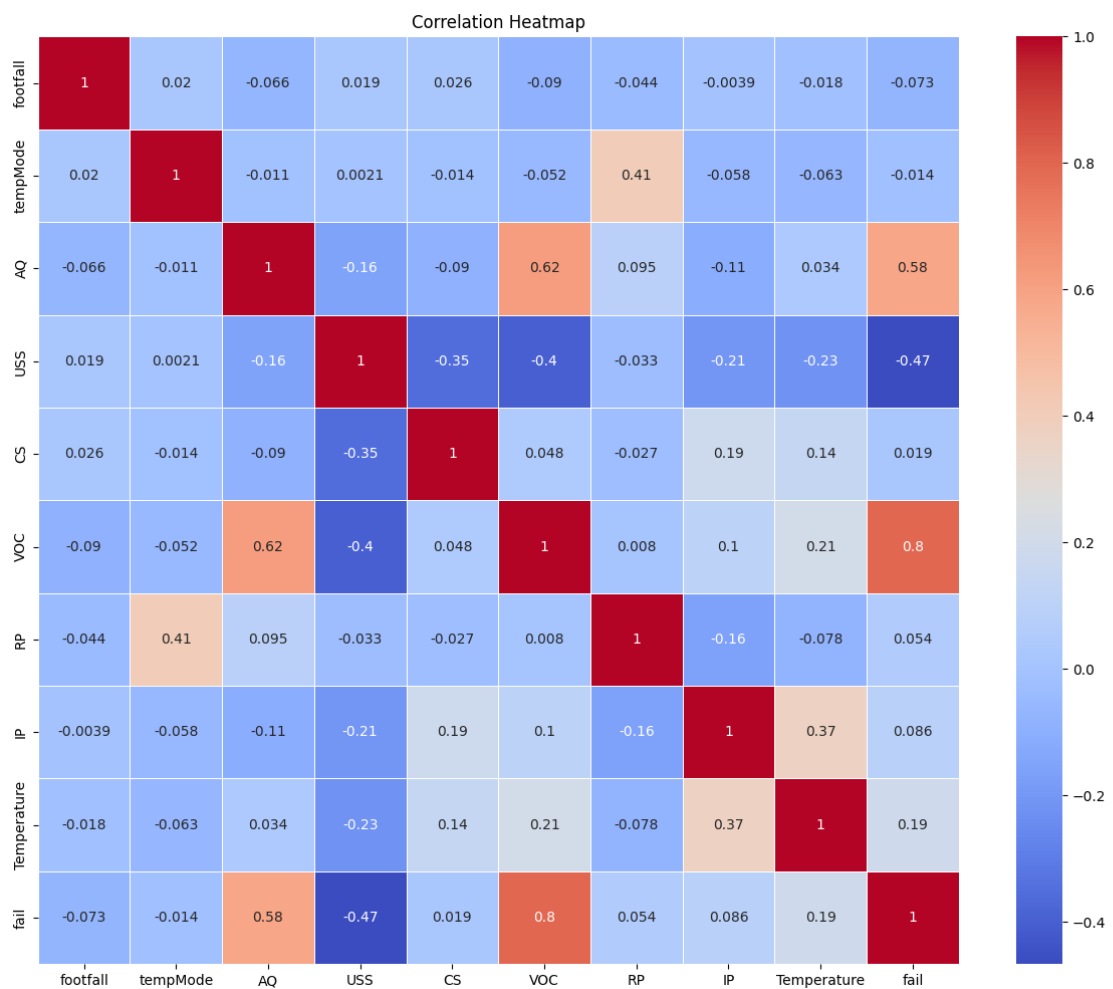
1) Confusion matrix: shows the model’s performance.



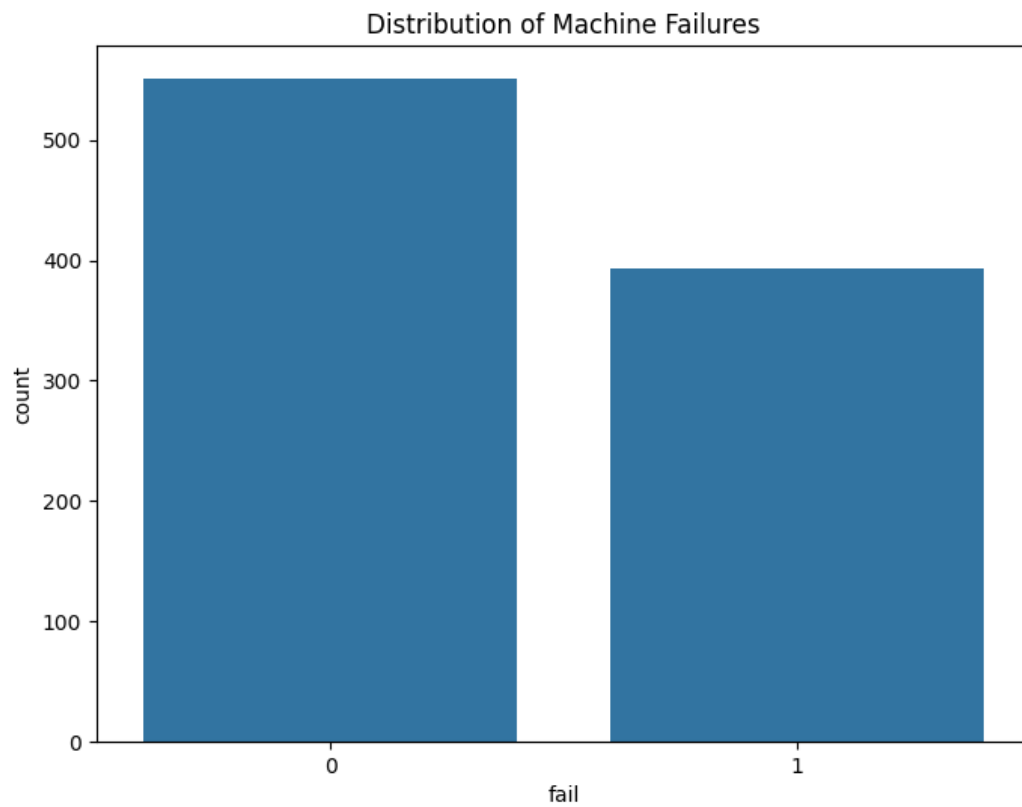
- True Negatives: 87, correctly predicted 87 instances where there was no machine failure.
- False Positives: 15, incorrectly predicted 15 instances as failures when they were actually non-failures.
- False Negatives: 10, missed 10 actual failures, predicting them as non-failures.
- True Positives: 77, correctly identified 77 instances of machine failures.

Model performed well in accurately in identifying failures and non- failures.

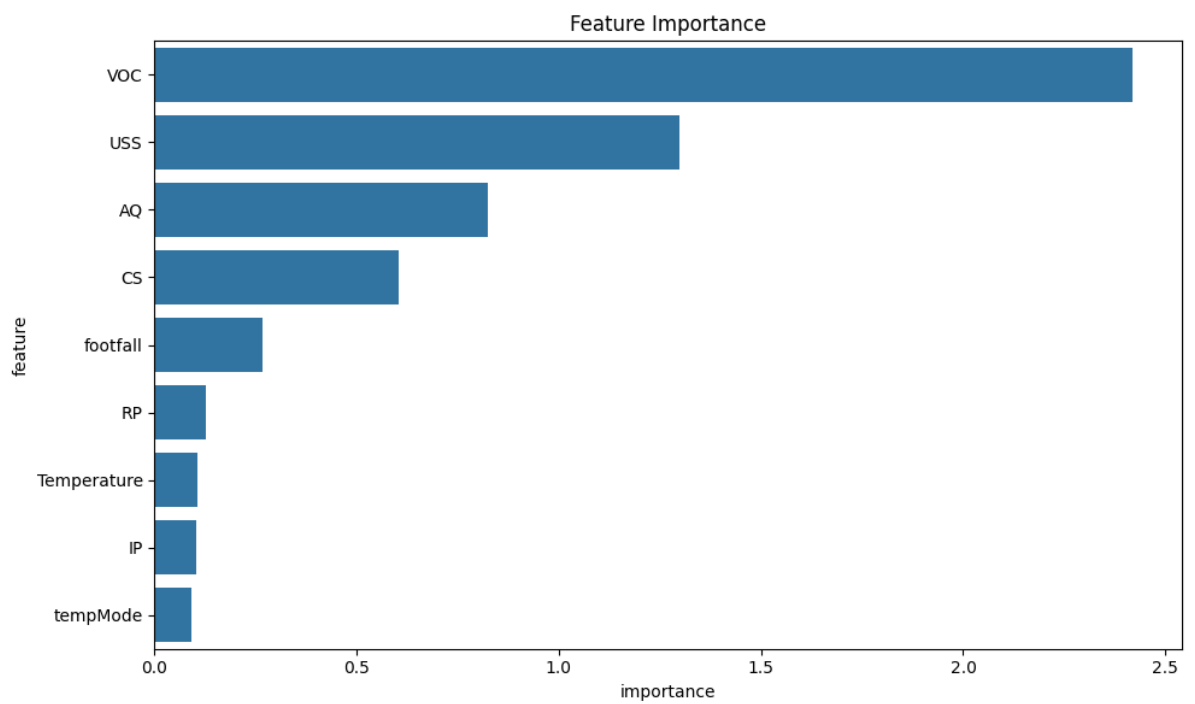
2) Correlation Heatmap: shows relationship between different features.



3) Failure Distribution: shows the distribution of machine failures in the dataset.



- 4) Feature Importance: highlights the most important features in predicting machine failures.



Other Insights and Results:

- 1) Top 5 most important features:

VOC 2.420079

USS 1.297031
AQ 0.825277
CS 0.603889
footfall 0.266227

2) ROC AUC Score: 0.9446

3) Classification Report:

	precision	recall	F1 - score	support
0	0.90	0.85	0.87	102
1	0.84	0.89	0.86	87
accuracy			0.87	189
macro avg	0.87	0.87	0.87	189
weight avg	0.87	0.87	0.87	189

4) Class distribution:

fail

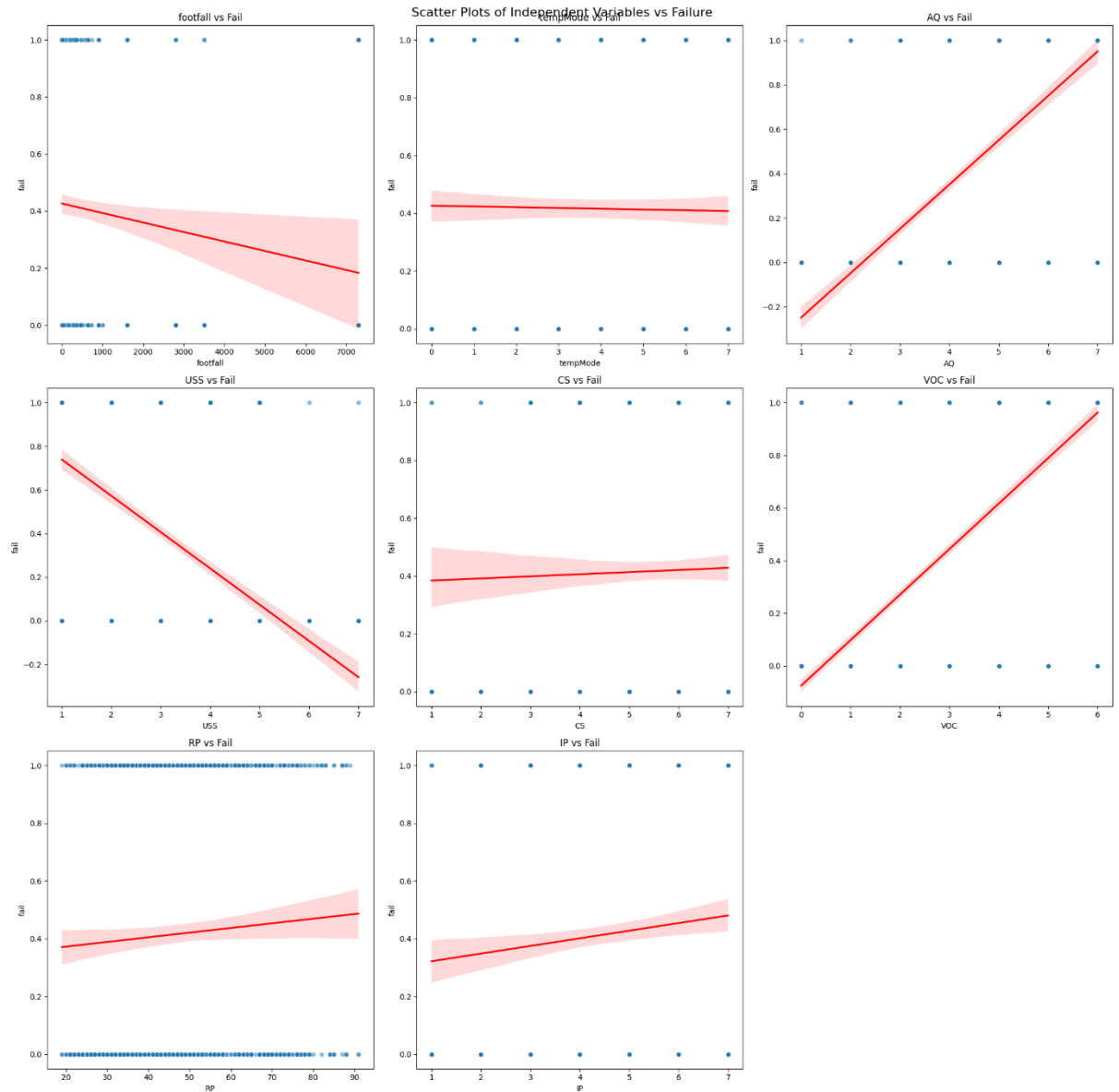
0 0.583686

1 0.416314

Name: proportion, dtype: float64

The logistic regression model performed well with a high ROC AUC score, indicating good predictive power. The most important features were VOC, USS, and AQ.

Correlation scatter plots to show the relationship between all features with the dependent variable (fail):



These correlation coefficients show the following observations from the scatter plots:

1. VOC has the strongest positive correlation (0.797) with failure.
2. AQ has the second strongest positive correlation (0.583).
3. USS has a moderate negative correlation (-0.467).
4. Temperature has a weak positive correlation (0.190).
5. Other variables have very weak correlations, either positive or negative.