CS-5710: Machine Learning

PREDICTIVE MAINTENANCE FOR MACHINE TOOL FAILURES

TEAMMATES:

GUTHIKONDA SAI PRANITHA -700740867

BOPPANA VEERA VENKATA

SATYANARAYANA -700740862

TEJASWI REDDY ANAPALLI -700740567

RAKESH PEDDAPALLI - 700740294

TITLE OF THE PROJECT: PREDICTIVE MAINTENANCE FOR MACHINE TOOL FAILURES

ABSTRACT:

In this project, we would like to fit the best model which can predict both when a failure will occur and the type of that failure. The independent variables provided in the dataset are Air temperature [k], Type, Process temperature [k], Rotational speed [rpm], Torque [Nm], Tool wear [min] which defines the failure type and when a failure occurs. Before applying the classification techniques, we will perform the Exploratory Data Analysis and feature selection such as checking for missing values, finding the correlation between the variables and visualizing the data.

The obtained data will be split into train and test splits. Then we fit the training dataset into classification models such as Logistic Regression, K-Nearest Neighbor (KNN), SVC, Random Forest. From all the classification algorithms trained, the best model with better accuracy is selected. For those models the confusion matrix will be displayed which tells the true positive and true negative rate and displays the precision, recall for all the models.

PREDICTIVE MAINTENANCE

In predictive maintenance, data is collected over time to monitor the state of equipment. The goal is to find patterns that can help predict and ultimately prevent failures.

Problem Statement

i. Predicting when a failure will occur using independent variables.

Predictor Variables: Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], Tool ware [min].

Response Variables: Target

ii. Predicting type of failure will occur using predicting variables.

Predictor Variables: Air temperature [k], Type, Process temperature [k], Rotational Speed [rpm], Torque [Nm], Tool ware [min].

Response Variables: Failure Type.

Dataset: For our scenario, the dataset contains 10 columns and 10000 entries in which there are three object, four int and three float datatype columns. Following is a screenshot of the dataset.

UDI	Product ID	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
1	M14860	M	298.1	308.6	1551	42.8	0	0	No Failure
2	L47181	L	298.2	308.7	1408	46.3	3	0	No Failure
3	L47182	L	298.1	308.5	1498	49.4	5	0	No Failure
4	L47183	L	298.2	308.6	1433	39.5	7	0	No Failure
5	L47184	L	298.2	308.7	1408	40	9	0	No Failure
6	M14865	M	298.1	308.6	1425	41.9	11	0	No Failure
7	L47186	L	298.1	308.6	1558	42.4	14	0	No Failure

Product Id contains different variants of tools and Type consists of Low(L), Medium(M), and High(H). Failure Type contains No Failure, Heat Dissipation Failure, Over Strain Failure, Power Failure, Random Failure, Tool Wear Failure and Target contains 0 and 1 where 0 denotes No Failure and 1 denotes any one of the Failures.

Predictive maintenance dataset link

Problem Solving Methodologies

3.1 Data Cleaning:

Before Predicting both when a failure will occur and the type of failure, we are going to clean the data applying exploratory data analysis to the given dataset. Below are the steps to be followed:

1. Dropping the unwanted features
2. Visualizing the data
3. Encoding
4. Scaling

3.1.1 Dropping the unwanted features:

Initially, we are going to drop the unwanted features from the dataset using drop () function. Dropped the features in order to achieve better accuracy.

```
data.drop(columns=['Product ID','UDI'],inplace=True,axis=1)
```

```
data.head()
```

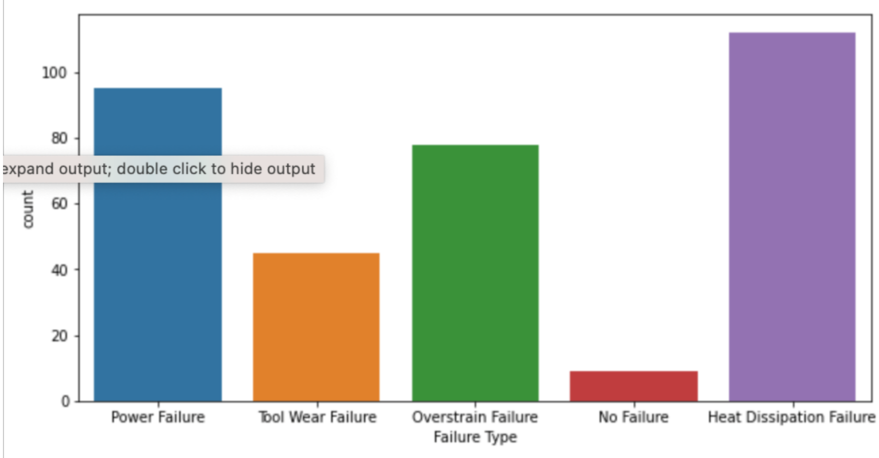
	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	M	298.1	308.6	1551	42.8	0	0	No Failure
1	L	298.2	308.7	1408	46.3	3	0	No Failure
2	L	298.1	308.5	1498	49.4	5	0	No Failure
3	L	298.2	308.6	1433	39.5	7	0	No Failure
4	L	298.2	308.7	1408	40.0	9	0	No Failure

3.1.2 Visualizing the data:

Here we used the countplot() function to visualize the failure type column.

```
plt.figure(figsize=(10,5))
sns.countplot(data=data[data['Target']==1],x="Failure Type")
```

```
<AxesSubplot:xlabel='Failure Type', ylabel='count'>
```



3.1.3 Encoding:

Using label_encoder_transform() function encoded the columns Type and Failure Type. Lets consider a column Failure Type which has 6 different entries such as Heat Dissipation Failure, No Failure, Overstrain Failure, Power Failure, Random Failure and Tool Wear Failure. By using label_encoder_transform() function the entry values are assigned as below:

Heat Dissipation Failure - 0

No Failure - 1

Over Strain Failure – 2

Power Failure –3

Random Failure – 4

Tool Wear Failure – 5

```
label=LabelEncoder()
data['Type']=label.fit_transform(data['Type'])
data['Failure Type']=label.fit_transform(data['Failure Type'])
```

```
data['Failure Type'].unique()
```

```
array([1, 3, 5, 2, 4, 0])
```

```
data.head()
```

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	2	298.1	308.6	1551	42.8	0	0	1
1	1	298.2	308.7	1408	46.3	3	0	1
2	1	298.1	308.5	1498	49.4	5	0	1
3	1	298.2	308.6	1433	39.5	7	0	1
4	1	298.2	308.7	1408	40.0	9	0	1

3.1.4 Scaling:

By using Minmax Scale method, Scaled the columns Air temperature [k], Type, Process temperature[k], Rotational speed [rpm], Torque [Nm], Tool wear [min] values between 0 to 1.

```
scaler = MinMaxScaler()
data[['Air temperature [K]','Process temperature [K]','Rotational speed [rpm]','Torque [Nm]'
```

```
data.head()
```

	Type	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Target	Failure Type
0	2	0.304348	0.358025	0.222934	0.535714	0.000000	0	1
1	1	0.315217	0.370370	0.139697	0.583791	0.011858	0	1
2	1	0.304348	0.345679	0.192084	0.626374	0.019763	0	1
3	1	0.315217	0.358025	0.154249	0.490385	0.027668	0	1
4	1	0.315217	0.370370	0.139697	0.497253	0.035573	0	1

4. Problem Statement 1 Classification models:

4.1 Test and Train Split:

After performing above data operations, the next step is to split the data into the train and test using train_test_split() function. We split the data with test size 0.3 and random_state as 746 in both cases.

```
x=data[['Type','Air temperature [K]','Process temperature [K]','Rotational speed [rpm]','Tor
y=data['Target']
```

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=746)
```

4.2 Proposed Classification Models:

4.2.1 Logistic Regression:

As the model tried to predict the outcome with test data accuracy score is 96.9%. Targets Precision and Recall score for class 0 is 0.97 and 1.00 and for class 1 is 1.00 and 0.01. f1_score is high for the class 0 I.e., 0.98.

```
lgr1=LogisticRegression()
lgr1.fit(x_train,y_train)
y_pred_lgr1=lgr1.predict(x_test)
```

```
accuracy_scr = round(accuracy_score(y_test,y_pred_lgr1), 3)
print("Accuracy Score of logistic regression [Target] : ", accuracy_scr)
print("Confusion Matrix of logistic regression [Target] :\n",confusion_matrix(y_test,y_pred_
print("Classification Report of logistic regression[Target] : \n",classification_report(y_te

cohen3_lgr1 = metrics.cohen_kappa_score(y_test,y_pred_lgr1)
print('Cohen Kappa: %.3f' % cohen3_lgr1)
```

```
Accuracy Score of logistic regression [Target] :  0.969
Confusion Matrix of logistic regression [Target] :
 [[2905    0]
 [  94    1]]
Classification Report of logistic regression[Target] :
               precision    recall  f1-score   support

           0       0.97      1.00      0.98      2905
           1       1.00      0.01      0.02        95

    accuracy                           0.97      3000
   macro avg       0.98      0.51      0.50      3000
weighted avg       0.97      0.97      0.95      3000

Cohen Kappa: 0.020
```

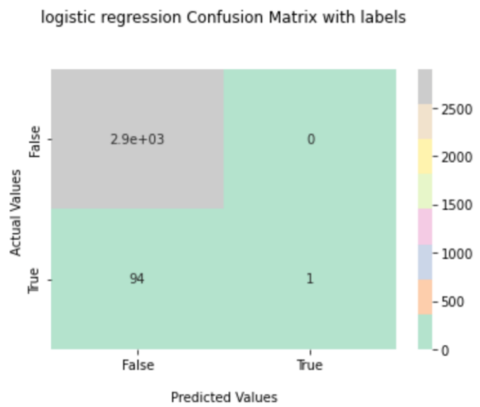
Model/Metrics	Accuracy Score	Confusion Matrix (Target)		Precision		Recall	
				Class Target (0)	Class Target (1)	Class Target (0)	Class Target (1)
Logistic Regression	96.9%	2905	0	0.97	1.00	1.00	0.01
		94	1				

```
ax = sns.heatmap(confusion_matrix(y_test,y_pred_lgr1), annot=True, cmap='Pastel2')

ax.set_title('logistic regression Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
[Text(0, 0.5, 'False'), Text(0, 1.5, 'True')]
```



Note: For the rest 3 models in our model, we have used GridSearchCV for hyper parameter tuning where in a range of parameters GridSearchCV finds the best parameters where the model performs well.

The kappa statistic is a measure of how closely the instances classified by the machine learning classifier matched the data labeled as ground truth, controlling for the accuracy of a random classifier as measured by the expected accuracy.

4.2.2 SVC Using GridSearchCV:

As the model tried to predict the outcome with test data accuracy score is 97.5%. Target's Precision and Recall score for class 0 is 0.97 and 1.00 and for class 1 is 1.00 and 0.21. f1_score is high for the class 0 i.e 0.99.

```
print("Best params from Support Vector Classifier [Target] : ",SVC_grid1.best_params_)
y_pred_SVC1=SVC_grid1.predict(x_test)
print("Accuracy Score of Support Vector Classifier [Target] : ",accuracy_score(y_test,y_pred
print("Confusion Matrix of Support Vector Classifier [Target] :\n",confusion_matrix(y_test,y
print("Classification Report of Support Vector Classifier[Target] : \n",classification_repor

cohen3_SVC1 = metrics.cohen_kappa_score(y_test,y_pred_SVC1)
print('Cohen Kappa: %.3f' % cohen3_SVC1)
```

```
Best params from Support Vector Classifier [Target] :  {'C': 10, 'degree': 1, 'kernel': 'rb
f'}
Accuracy Score of Support Vector Classifier [Target] :  0.975
Confusion Matrix of Support Vector Classifier [Target] :
 [[2905    0]
 [  75   20]]
Classification Report of Support Vector Classifier[Target] :
               precision    recall  f1-score   support

           0       0.97      1.00      0.99      2905
           1       1.00      0.21      0.35        95

    accuracy                           0.97      3000
   macro avg       0.99      0.61      0.67      3000
weighted avg       0.98      0.97      0.97      3000

Cohen Kappa: 0.341
```

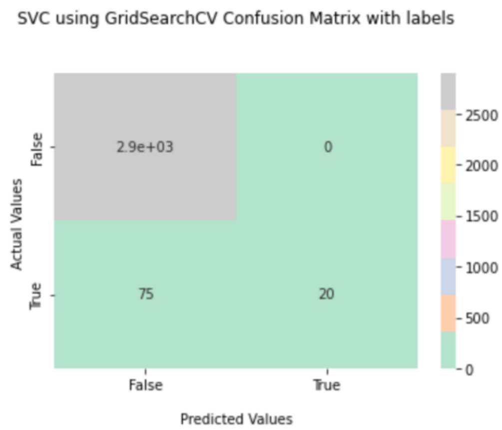
<table>
<tr><th rowspan="2">Model/Metrics</th><th rowspan="2">Accuracy Score</th><th rowspan="2" colspan="2">Confusion Matrix (Target)</th><th colspan="2">Precision</th><th colspan="2">Recall</th></tr>
<tr><th>Class Target (0)</th><th>Class Target (1)</th><th>Class Target (0)</th><th>Class Target (1)</th></tr>
<tr><td rowspan="2">SVC</td><td rowspan="2">97.5%</td><td>2905</td><td>0</td><td rowspan="2">0.97</td><td rowspan="2">1.00</td><td rowspan="2">1.00</td><td rowspan="2">0.21</td></tr>
<tr><td>75</td><td>20</td></tr>
</table>

```
ax = sns.heatmap(confusion_matrix(y_test,y_pred_SVC1), annot=True, cmap='Pastel2')

ax.set_title('SVC using GridSearchCV Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
[Text(0, 0.5, 'False'), Text(0, 1.5, 'True')]
```



4.2.3 Random Forest Using Grid Search CV:

As the model tried to predict the outcome with test data accuracy score is 98.0%. Target's Precision and Recall score for class 0 is 0.98 and 1.00 and for class 1 is 0.91 and 0.41. f1_score is high for the class 0 i.e 0.99.

```
print("Best params from Random Forest Classifier [Target] : ",rf_grid1.best_params_)
y_pred_rf1=rf_grid1.predict(x_test)
print("Accuracy Score of Random Forest Classifier [Target] : ",accuracy_score(y_test,y_pred_r
print("Confusion Matrix of Random Forest Classifier [Target] :\n",confusion_matrix(y_test,y_p
print("Classification Report of Random Forest Classifier[Target] : \n",classification_report(

cohen3_rf1 = metrics.cohen_kappa_score(y_test,y_pred_rf1)
print('Cohen Kappa: %.3f' % cohen3_rf1)
```

```
Best params from Random Forest Classifier [Target] :  {'criterion': 'entropy', 'max_depth':
8}
Accuracy Score of Random Forest Classifier [Target] :  0.98
Confusion Matrix of Random Forest Classifier [Target] :
 [[2901    4]
 [  56   39]]
Classification Report of Random Forest Classifier[Target] :
               precision    recall  f1-score   support

           0       0.98      1.00      0.99      2905
           1       0.91      0.41      0.57        95

    accuracy                           0.98      3000
   macro avg       0.94      0.70      0.78      3000
weighted avg       0.98      0.98      0.98      3000

Cohen Kappa: 0.556
```

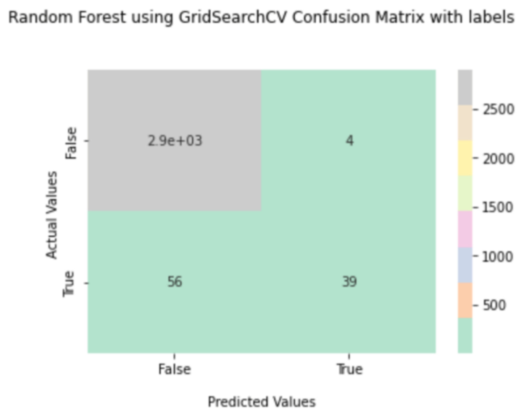
Model/Metrics	Accuracy Score	Confusion Matrix (Target)		Precision		Recall	
				Class Target (0)	Class Target (1)	Class Target (0)	Class Target (1)
Random Forest	98.0%	2901	4	0.98	1.00	0.91	0.41
		56	39				

```
ax = sns.heatmap(confusion_matrix(y_test,y_pred_rf1), annot=True, cmap='Pastel2')

ax.set_title('Random Forest using GridSearchCV Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
[Text(0, 0.5, 'False'), Text(0, 1.5, 'True')]
```



4.2.4 K-Nearest Neighbor:

As the model tried to predict the outcome with test data accuracy score is 97.6%. Target's Precision and Recall score for class 0 is 0.98 and 1.00 and for class 1 is 0.77 and 0.35. f1_score is high for the class 0 i.e 0.99.

```
print("Best params from K- Nearest Neighbor [Target] : ",knn_grid1.best_params_)
y_pred_knn1=knn_grid1.predict(x_test)
print("Accuracy Score of K- Nearest Neighbor [Target] : ",accuracy_score(y_test,y_pred_knn1)
print("Confusion Matrix of K- Nearest Neighbor [Target] :\n",confusion_matrix(y_test,y_pred_
print("Classification Report of K- Nearest NeighborTarget] : \n",classification_report(y_tes

cohen3_knn1 = metrics.cohen_kappa_score(y_test,y_pred_knn1)
print('Cohen Kappa: %.3f' % cohen3_knn1)
```

```
Best params from K- Nearest Neighbor [Target] :  {'n_neighbors': 3}
Accuracy Score of K- Nearest Neighbor [Target] :  0.976
Confusion Matrix of K- Nearest Neighbor [Target] :
 [[2895   10]
 [  62   33]]
Classification Report of K- Nearest NeighborTarget] :
               precision    recall  f1-score   support

           0       0.98      1.00      0.99      2905
           1       0.77      0.35      0.48        95

    accuracy                           0.98      3000
   macro avg       0.87      0.67      0.73      3000
weighted avg       0.97      0.98      0.97      3000

Cohen Kappa: 0.468
```

<table>
<tr><th rowspan="2">Model/Metrics</th><th rowspan="2">Accuracy Score</th><th rowspan="2" colspan="2">Confusion Matrix (Target)</th><th colspan="2">Precision</th><th colspan="2">Recall</th></tr>
<tr><th>Class Target (0)</th><th>Class Target (1)</th><th>Class Target (0)</th><th>Class Target (1)</th></tr>
<tr><td rowspan="2">K-Nearest Neighbor</td><td rowspan="2">97.6%</td><td>2895</td><td>10</td><td rowspan="2">0.98</td><td rowspan="2">0.77</td><td rowspan="2">1.00</td><td rowspan="2">0.35</td></tr>
<tr><td>62</td><td>33</td></tr>
</table>

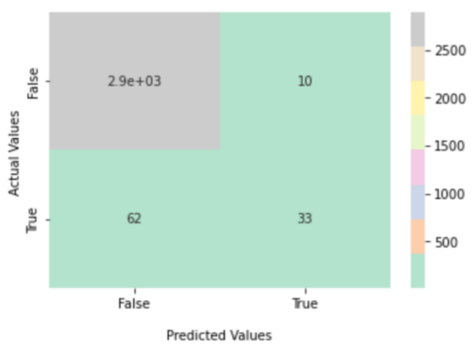
```
ax = sns.heatmap(confusion_matrix(y_test,y_pred_knn1), annot=True, cmap='Pastel2')


ax.set_title('K-nearest Neighbor using GridSearchCV Confusion Matrix with labels\n\n');
ax.set_xlabel('\nPredicted Values')
ax.set_ylabel('Actual Values ');

## Ticket labels - List must be in alphabetical order
ax.xaxis.set_ticklabels(['False','True'])
ax.yaxis.set_ticklabels(['False','True'])
```

```
[Text(0, 0.5, 'False'), Text(0, 1.5, 'True')]
```

K-nearest Neighbor using GridSearchCV Confusion Matrix with labels



Implementation Status Report:

Work Completed:

Performed Data cleaning on the dataset and completed problem statement 1 using Linear Regression, SVC algorithm, Random Forest and K-Nearest Neighbor.

Responsibility:

Divided the classification algorithms among us.

Linear Regression: Rakesh Peddapalli

SVC: Boppana Veera Venkata Satyanarayana

Random Forest: Guthikonda Sai Pranitha

K-Nearest Neighbor: Tejaswi Reddy Anapalli

Group Work: We performed data cleaning together and documented the report.

Work to be completed:

Need to complete problem statement 2.

Responsibility:

Divided the classification algorithms among us.

Linear Regression: Rakesh Peddapalli

SVC: Boppana Veera Venkata Satyanarayana

Random Forest: Guthikonda Sai Pranitha

K-Nearest Neighbor: Tejaswi Reddy Anapalli

Concerns/Issues: No issues upto date

References:

1. Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, Symone G. S. Alcalá,
   A systematic literature review of machine learning methods applied to predictive maintenance
   A systematic literature review of machine learning methods applied to predictive maintenance - ScienceDirect
2. M. Paolanti, L. Romeo, A. Felicetti, A. Mancini, E. Frontoni and J. Loncarski, "Machine Learning approach for Predictive Maintenance in Industry 4.0," *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*, 2018, pp. 1-6, doi: 10.1109/MESA.2018.8449150.
   Machine Learning approach for Predictive Maintenance in Industry 4.0 | IEEE Conference Publication | IEEE Xplore
3. L. Bin and Y. Min, "Analysis Model of Drilling Tool Failure Based on PSO-SVM and Its Application," 2012 Fourth International Conference on Computational and Information Sciences, 2012, pp. 1307-1310, doi: 10.1109/ICCIS.2012.75.
   Analysis Model of Drilling Tool Failure Based on PSO-SVM and Its Application | IEEE Conference Publication | IEEE Xplore
4. Sohyung Cho, Shihab Asfour, Arzu Onar, Nandita Kaundinya,
   Tool breakage detection using support vector machine learning in a milling process,
   International Journal of Machine Tools and Manufacture
   Tool breakage detection using support vector machine learning in a milling process - ScienceDirect