Final_Report

109550039 楊富翔

Github
Weight

I. Preprocessing

1. Fill the lost data (reference by link1)

```
imputer = SimpleImputer(strategy='mean')
imputer.fit(train_data[numerical_cols].append(test_data[numerical_cols]))
train_data[numerical_cols] = imputer.transform(train_data[numerical_cols])
```

Use SimpleImputer from sklearn to fill the missing value with mean

✓ Fit function contains test_data can get higher score since model take test into consideration

2. Transform non_numerical feature

```
for column in categorical_cols:
    label_encoder = LabelEncoder()
    label_encoder.fit(train_data[column].append(test_data[column]))
    train_data[column] = label_encoder.transform(train_data[column])
train_data.head()
```

Use LabelEncoder from sklearn to transform [‘product_code’, ‘attribute_0’ , ‘attribute_1’] into interger

✓ Fit function contains test_data can get higher score since model take test into consideration

3. Select features

A. Feature Engineering (reference by link2, link3)

```
train_data['attribute_2*3'] = train_data['attribute_2'] * train_data['attribute_3']
numerical_cols = numerical_cols + ['attribute_2*3']
```

From link2, I multiply attribute_2&attribute_3 since attribute0~3 are fixed for same product_code and train_data is A~E, test_data is F~I, so it’s useful to determine product_code.

```
meas_gr1_cols = [f"measurement_{i:d}" for i in list(range(3, 4)) +list(range(5, 7))+ list(range(8, 9))]
train_data['meas_gr1_avg'] = np.mean(train_data[meas_gr1_cols], axis=1)
numerical_cols = numerical_cols + ['meas_gr1_avg']
train_data['meas_gr1_std'] = np.std(train_data[meas_gr1_cols], axis=1)
numerical_cols = numerical_cols + ['meas_gr1_std']
```

```
meas_gr2_cols = [f"measurement_{i:d}" for i in list(range(7, 8))+list(range(9, 10))+list(range(13, 14))]
train_data['meas_gr2_avg'] = np.mean(train_data[meas_gr2_cols], axis=1)
numerical_cols = numerical_cols + ['meas_gr2_avg']
train_data['meas_gr2_std'] = np.std(train_data[meas_gr2_cols], axis=1)
numerical_cols = numerical_cols + ['meas_gr2_std']
```

Also from link2 aggregate measurement_3 to 16 into average and stdev, I split it to 2 aggregate by higher value and lower value and remove data with higher std from train_data.describe()

```
# train_data['m_3_missing'] = train_data.measurement_3.isna()
# train_data['m_5_missing'] = train_data.measurement_5.isna()
```

From link3, missing is significant feature to get score 58, but it'll decrease my score when I get score 59. I abort it finally.

B. mutual_info_classif(reference by link4)

```
mi_scores = mutual_info_classif(X, y, random_state=1)
mi_scores = pd.Series(mi_scores, name="MI Scores")
mi_scores_classif = mi_scores.sort_values(ascending=False)
```

Use mutual_info_classif from sklearn to get positive correlative features

II. Train

1. Modeling

```
model = LogisticRegression(C=0.0001, penalty='l2', solver='newton-cg')
model.fit(X,y)
predictions = model.predict_proba(X)[:, 1]
score = roc_auc_score(y, predictions)
print(f"AUC: {score}")

#save model
joblib.dump(model, '109550039_model')
```

Use LogisticRegression and test hyperparameters from

'lbfgs'	['l2', None]
'liblinear'	['l1', 'l2']
'newton-cg'	['l2', None]
'newton-cholesky'	['l2', None]
'sag'	['l2', None]
'saga'	['elasticnet', 'l1', 'l2', None]

Get 'l2', 'newton-cg' is the best

III. Inference

1. Preprocessing

   Same as training

2. Load model and Preditct

```
model = joblib.load('109550039_model')
sub['failure'] = model.predict_proba(test_X)[:, 1]
sub.to_csv('109550039.csv', index=False)
```

✓ test_X should only contain the colums from training

3. Result

reference code

[code 1]

[code 2]

[code 3]