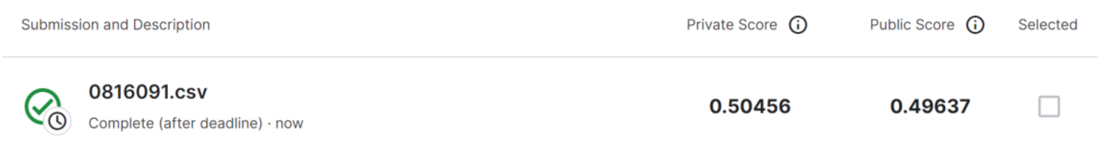
Link of github:
https://github.com/ginnyching/2022-Fall-Machine-Learning/tree/main/ML_Final_Project

Screenshot on Kaggle:

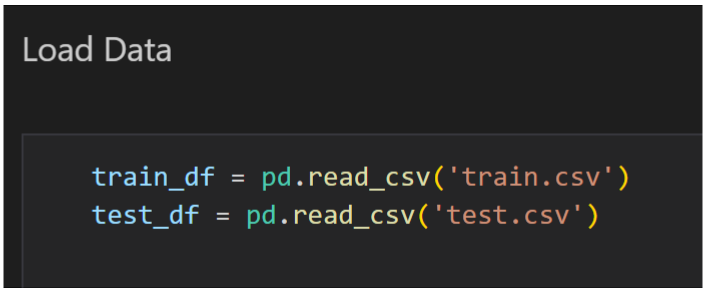


Submission and Description	Private Score	Public Score	Selected
0816091.csv Complete (after deadline) · now	0.50456	0.49637	☐

參考資料:
https://github.com/mohamedali-sc/Tabular-Playground-Series-Aug-2022

- Inference:

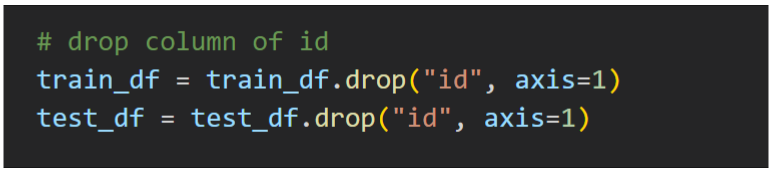
1. Load data from csv file



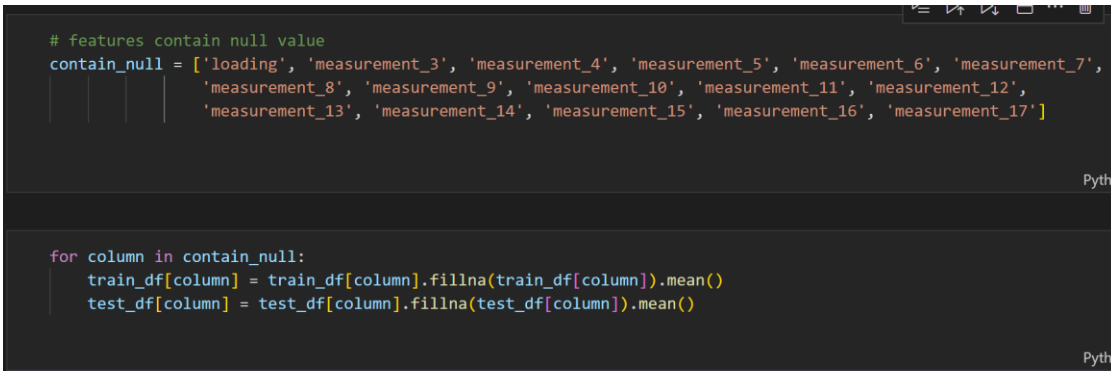
Load Data

```python
train_df = pd.read_csv('train.csv')
test_df = pd.read_csv('test.csv')
```

2. Drop unnecessary column and fill null with its mean



```python
# drop column of id
train_df = train_df.drop("id", axis=1)
test_df = test_df.drop("id", axis=1)
```



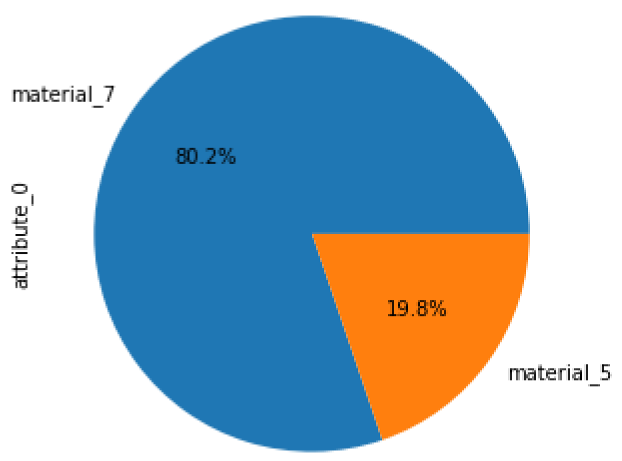
```python
# features contain null value
contain_null = ['loading', 'measurement_3', 'measurement_4', 'measurement_5', 'measurement_6', 'measurement_7',
                'measurement_8', 'measurement_9', 'measurement_10', 'measurement_11', 'measurement_12',
                'measurement_13', 'measurement_14', 'measurement_15', 'measurement_16', 'measurement_17']
```

```python
for column in contain_null:
    train_df[column] = train_df[column].fillna(train_df[column]).mean()
    test_df[column] = test_df[column].fillna(test_df[column]).mean()
```

3. plot pie chart to see the ratio of each features

```python
# pie chart according to each selected feature
for i, feature in enumerate(type_select):
    if i != 0:
        plt.subplots(1, 1, figsize=(5, 5))
        train_df[feature].value_counts().plot(kind='pie', autopct='%1.1f%%')
```

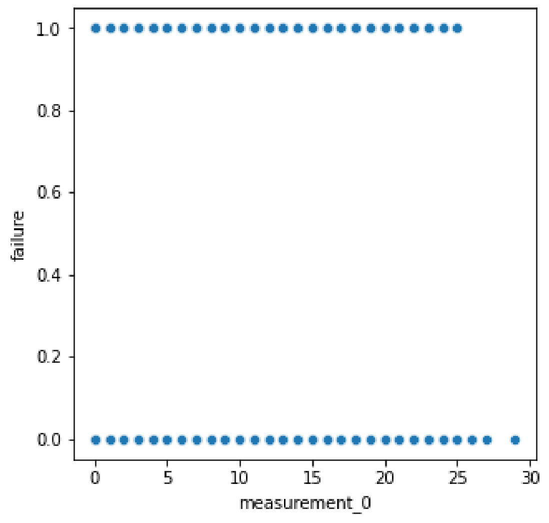
Example of pie chart using attribute_0:



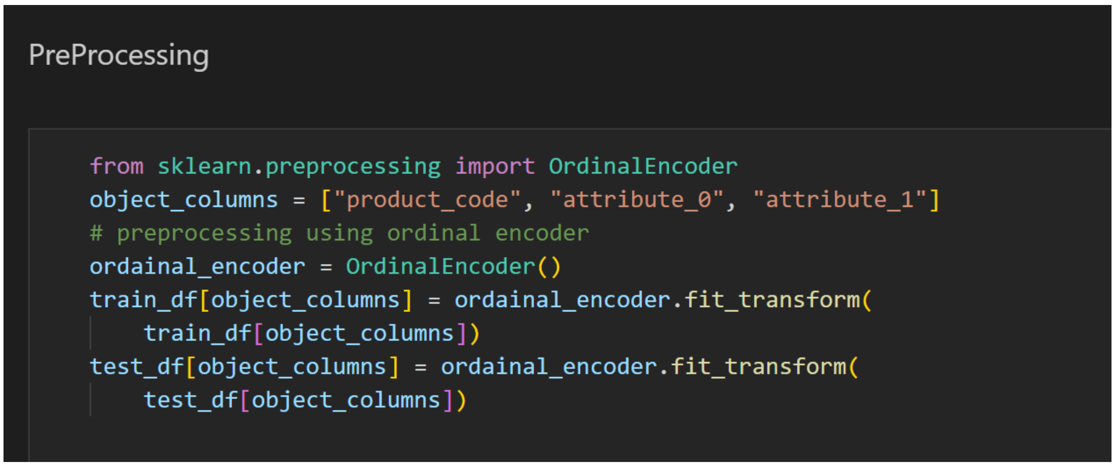
4. plot scatter plot to see how it relates to failure:

```python
# scatter plot of each numerical feature according to failure
plt.figure(figsize=(30, 60))
for i, feature in enumerate(numerics):
    plt.subplots(figsize=(5, 5))
    sns.scatterplot(x=feature, y="failure", data=train_df)
```

Example of scatter plot:



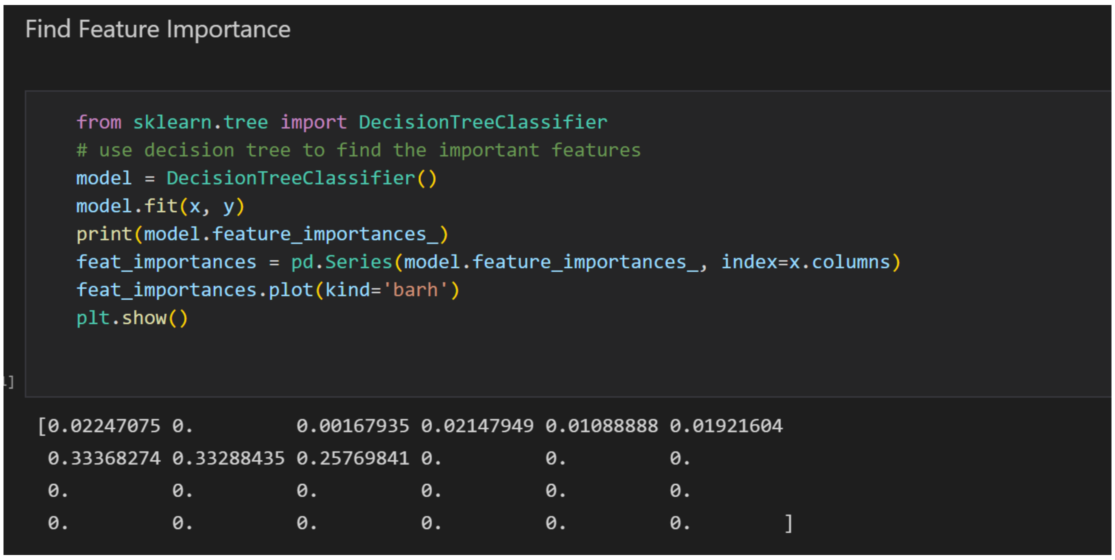
5. PreProcessing using Ordinal Encoder with object type columns:



PreProcessing

```python
from sklearn.preprocessing import OrdinalEncoder
object_columns = ["product_code", "attribute_0", "attribute_1"]
# preprocessing using ordinal encoder
ordainal_encoder = OrdinalEncoder()
train_df[object_columns] = ordainal_encoder.fit_transform(
    train_df[object_columns])
test_df[object_columns] = ordainal_encoder.fit_transform(
    test_df[object_columns])
```

6. Find feature importance using decision tree:

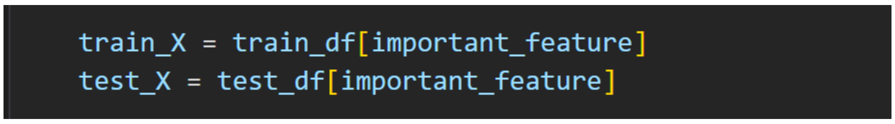


Find Feature Importance

```python
from sklearn.tree import DecisionTreeClassifier
# use decision tree to find the important features
model = DecisionTreeClassifier()
model.fit(x, y)
print(model.feature_importances_)
feat_importances = pd.Series(model.feature_importances_, index=x.columns)
feat_importances.plot(kind='barh')
plt.show()
```

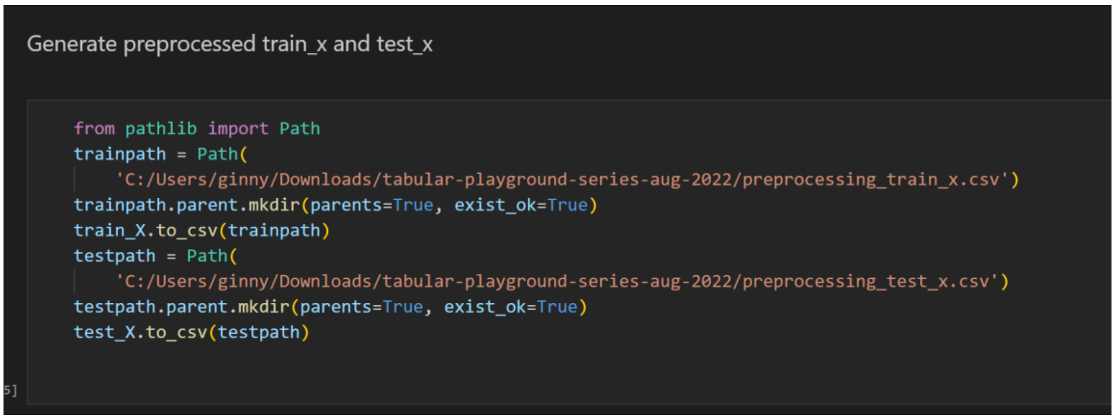
```
[0.02247075 0.         0.00167935 0.02147949 0.01088888 0.01921604
 0.33368274 0.33288435 0.25769841 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.        ]
```

7. Generate new train_X and test_X using important feature:



```python
train_X = train_df[important_feature]
test_X = test_df[important_feature]
```

8. Output the new preprocessing.csv

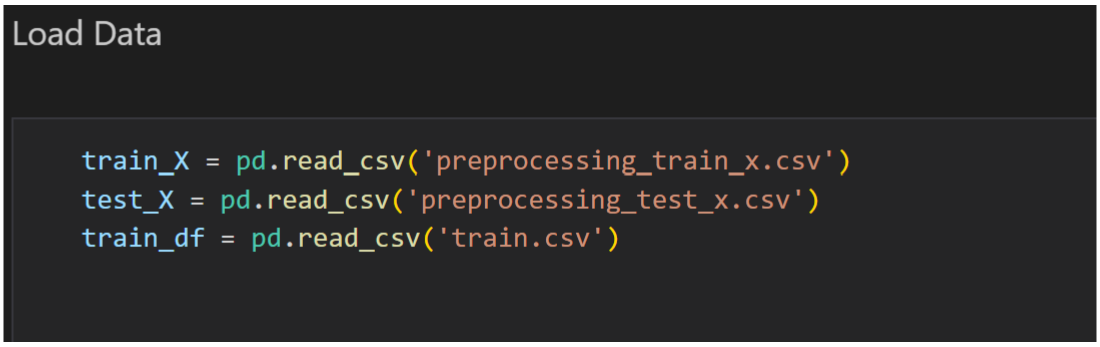


Generate preprocessed train_x and test_x

```python
from pathlib import Path
trainpath = Path(
    'C:/Users/ginny/Downloads/tabular-playground-series-aug-2022/preprocessing_train_x.csv')
trainpath.parent.mkdir(parents=True, exist_ok=True)
train_X.to_csv(trainpath)
testpath = Path(
    'C:/Users/ginny/Downloads/tabular-playground-series-aug-2022/preprocessing_test_x.csv')
testpath.parent.mkdir(parents=True, exist_ok=True)
test_X.to_csv(testpath)
```

- Training

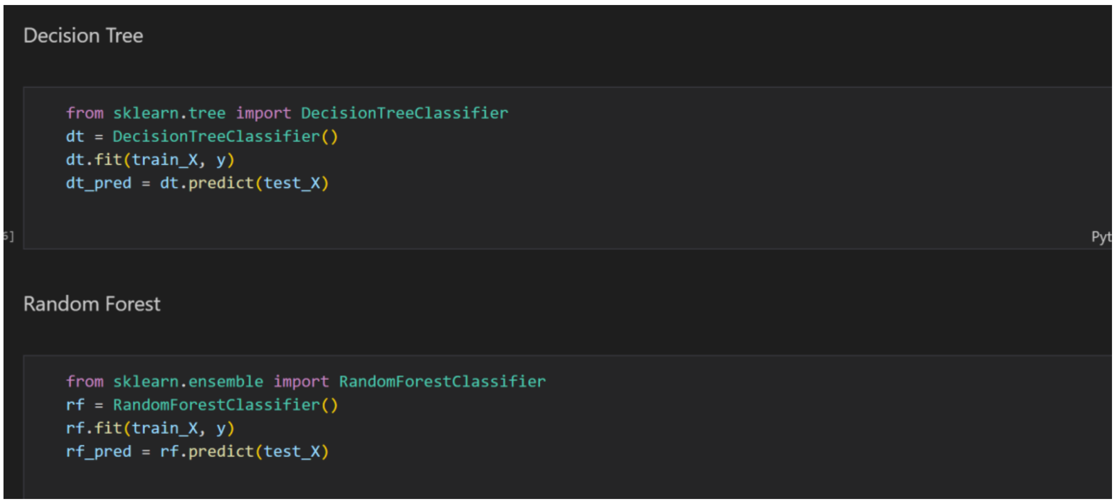
1. Load data from csv files



Load Data

```
train_X = pd.read_csv('preprocessing_train_x.csv')
test_X = pd.read_csv('preprocessing_test_x.csv')
train_df = pd.read_csv('train.csv')
```

2. use different methods to train and find the best method
   methods I used: Decision Tree, Random Forest, Histogram-based Gradient Boost, XGBoost, AdaBoost, LGBM, Ensemble(Stacking Classifier)
   example code of training:



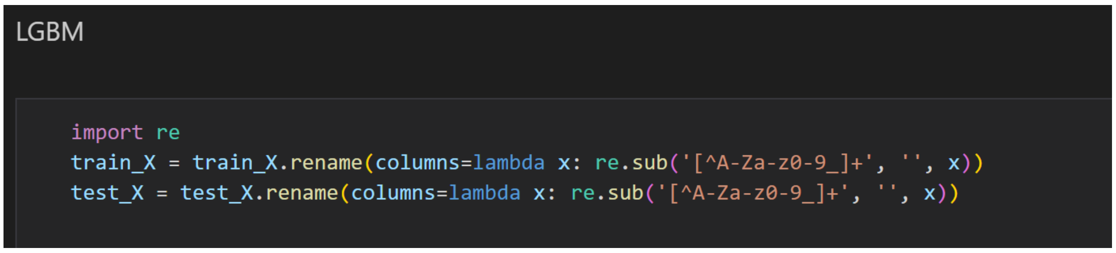
Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier()
dt.fit(train_X, y)
dt_pred = dt.predict(test_X)
```

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier()
rf.fit(train_X, y)
rf_pred = rf.predict(test_X)
```

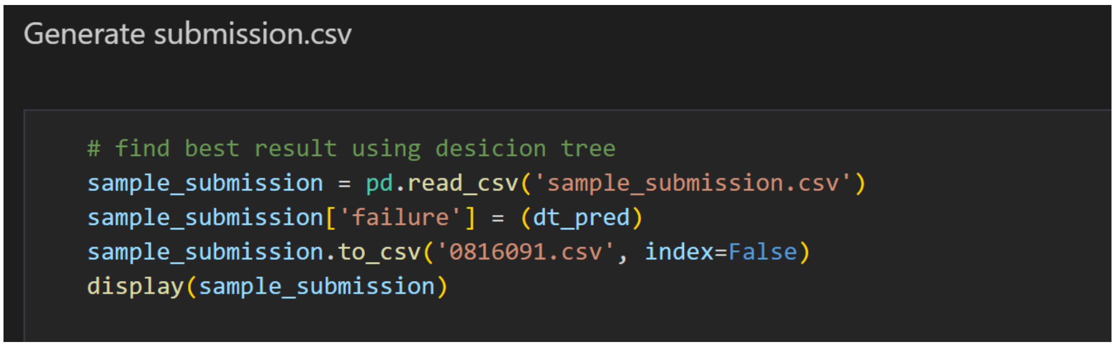
   Problem occurred at LGBM:
   Do not support special json characters in feature name.
   Solution: rename the feature name of train_X and test_X.



LGBM

```
import re
train_X = train_X.rename(columns=lambda x: re.sub('[^A-Za-z0-9_]+', '', x))
test_X = test_X.rename(columns=lambda x: re.sub('[^A-Za-z0-9_]+', '', x))
```

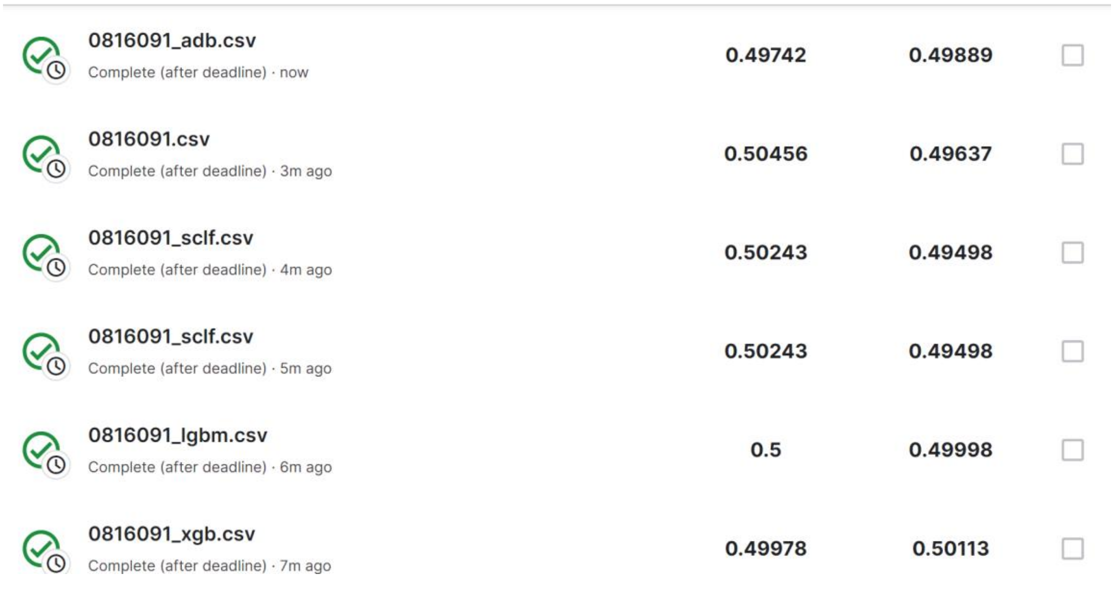
3. Found best method is Decision Tree, generate submission csv according to its prediction.



Generate submission.csv

```
# find best result using desicion tree
sample_submission = pd.read_csv('sample_submission.csv')
sample_submission['failure'] = (dt_pred)
sample_submission.to_csv('0816091.csv', index=False)
display(sample_submission)
```

4. Submit on Kaggle:



0816091_adb.csv Complete (after deadline) · now	0.49742	0.49889	☐
0816091.csv Complete (after deadline) · 3m ago	0.50456	0.49637	☐
0816091_sclf.csv Complete (after deadline) · 4m ago	0.50243	0.49498	☐
0816091_sclf.csv Complete (after deadline) · 5m ago	0.50243	0.49498	☐
0816091_lgbm.csv Complete (after deadline) · 6m ago	0.5	0.49998	☐
0816091_xgb.csv Complete (after deadline) · 7m ago	0.49978	0.50113	☐