Student name: Nguyen Anh Tuan
Student ID: 103805949

Swinburne University of Technology

COS40007-Artificial Intelligence for Engineering

Portfolio 3

Nguyen Anh Tuan - 103805949

Studio class: Studio 1-1

Hanoi, Vietnam

Student name: Nguyen Anh Tuan
Student ID: 103805949

1. Summary table of Studio 3 (Activity 6)

SVM model	Train-test split	Cross-validation
Original features	89.25%	89.18%
With hyperparameter tuning	84.47%	84.31%
With feature selection and hype parameter tuning	85.73%	84.31%
With PCA and hyper parameter tuning	84.61%	84.38%

2. Summary table of Studio 3 (Activity 7)

Model	Train-test split	Cross-validation
SGD	88.05%	87.84%
RandomForest	92.61%	92.43%
MLP	86.76%	87.82%
SVM	89.25%	89.18%

Student name: Nguyen Anh Tuan
Student ID: 103805949

3. Step 1: Data Collection

- Link: https://www.kaggle.com/code/twananguyen/submit-portfolio-3
- Data:
  - https://drive.google.com/drive/folders/1XGKEwtNActy6Gf1coKV3wqZeebFuV938?usp=sharing

STEP 1: DATA COLLECTION

In [2]:
```
boning = pd.read_csv('/kaggle/input/portfolio3/Boning.csv')
slicing = pd.read_csv('/kaggle/input/portfolio3/Slicing.csv')

boning['Class'] = 0
slicing['Class'] = 1

combined = pd.concat([boning, slicing], ignore_index=True)

selected_columns = ['Frame', 'L5 x', 'L5 y', 'L5 z',
                    'T12 x', 'T12 y', 'T12 z', 'Class']

combined = combined[selected_columns]
```

In [3]:
```
combined.head()
```

Out[3]:

	Frame	L5 x	L5 y	L5 z	T12 x	T12 y	T12 z	Class
0	0	0.052654	0.039386	-0.077002	0.099458	0.074396	-0.145448	0
1	1	-0.053525	0.117279	0.150245	-0.103594	0.226658	0.284348	0
2	2	0.073929	-0.022381	0.032701	0.137189	-0.036791	0.061717	0
3	3	-0.037295	-0.009975	0.015846	-0.028150	0.017239	0.061258	0
4	4	0.091745	-0.014404	0.054168	0.172983	-0.023023	0.102274	0

In [4]:
```
combined.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 72060 entries, 0 to 72059
Data columns (total 8 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Frame   72060 non-null  int64
 1   L5 x    72060 non-null  float64
 2   L5 y    72060 non-null  float64
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

4. Step 2: Create composite columns

- Link: https://www.kaggle.com/code/twananguyen/submit-portfolio-3
- Data:
    - https://drive.google.com/drive/folders/1XGKEwtNActy6Gf1coKV3wqZeebFuV938?usp=sharing

STEP 2: CREATE NEW COMPOSITE FEATURE

In [5]:

```
def computed_compo_feat(df):
    composite_feature = {}

    #  L5
    composite_feature['RMS_xy_L5'] = np.sqrt(df['L5 x']**2 + df['L5 y']**2)
    composite_feature['RMS_yz_L5'] = np.sqrt(df['L5 y']**2 + df['L5 z']**2)
    composite_feature['RMS_zx_L5'] = np.sqrt(df['L5 z']**2 + df['L5 x']**2)
    composite_feature['RMS_xyz_L5'] = np.sqrt(df['L5 x']**2 + df['L5 y']**2 + df['L5 z']**2)
    composite_feature['Roll_L5'] = 180 * np.arctan2(df['L5 y'], np.sqrt(df['L5 x']**2 + df['L5 z']**2)) / np.pi
    composite_feature['Pitch_L5'] = 180 * np.arctan2(df['L5 x'], np.sqrt(df['L5 y']**2 + df['L5 z']**2)) / np.pi

    # T12
    composite_feature['RMS_xy_T12'] = np.sqrt(df['T12 x']**2 + df['T12 y']**2)
    composite_feature['RMS_yz_T12'] = np.sqrt(df['T12 y']**2 + df['T12 z']**2)
    composite_feature['RMS_zx_T12'] = np.sqrt(df['T12 z']**2 + df['T12 x']**2)
    composite_feature['RMS_xyz_T12'] = np.sqrt(df['T12 x']**2 + df['T12 y']**2 + df['T12 z']**2)
```

```python
    # T12
    composite_feature['RMS_xy_T12'] = np.sqrt(df['T12 x']**2 + df['T12 y']**2)
    composite_feature['RMS_yz_T12'] = np.sqrt(df['T12 y']**2 + df['T12 z']**2)
    composite_feature['RMS_zx_T12'] = np.sqrt(df['T12 z']**2 + df['T12 x']**2)
    composite_feature['RMS_xyz_T12'] = np.sqrt(df['T12 x']**2 + df['T12 y']**2 + df['T
12 z']**2)
    composite_feature['Roll_T12'] = 180 * np.arctan2(df['T12 y'], np.sqrt(df['T12 x']*
*2 + df['T12 z']**2)) / np.pi
    composite_feature['Pitch_T12'] = 180 * np.arctan2(df['T12 x'], np.sqrt(df['T12 y']
**2 + df['T12 z']**2)) / np.pi

    # Update composite columns to DataFrame
    for key, value in composite_feature.items():
        df[key] = value

    return df

combined = computed_compo_feat(combined)

# Update order
order = ['Frame', 'L5 x', 'L5 y', 'L5 z', 'T12 x', 'T12 y', 'T12 z',
                 'RMS_xy_L5', 'RMS_yz_L5', 'RMS_zx_L5', 'RMS_xyz_L5', 'Roll_L5', 'Pitc
h_L5',
```


```python
    composite_feature['Pitch_T12'] = 180 * np.arctan2(df['T12 x'], np.sqrt(df['T12 y']
**2 + df['T12 z']**2)) / np.pi

    # Update composite columns to DataFrame
    for key, value in composite_feature.items():
        df[key] = value

    return df

combined = computed_compo_feat(combined)

# Update order
order = ['Frame', 'L5 x', 'L5 y', 'L5 z', 'T12 x', 'T12 y', 'T12 z',
                 'RMS_xy_L5', 'RMS_yz_L5', 'RMS_zx_L5', 'RMS_xyz_L5', 'Roll_L5', 'Pitc
h_L5',
```


```python
                 'RMS_xy_T12', 'RMS_yz_T12', 'RMS_zx_T12', 'RMS_xyz_T12', 'Roll_T12',
'Pitch_T12',
                 'Class']
combined = combined[order]
```

In [6]:

```python
combined.info()
```

5. Step 3: Data pre-processing

- Link: https://www.kaggle.com/code/twananguyen/submit-portfolio-3
- Data:

Student name: Nguyen Anh Tuan
Student ID: 103805949

- https://drive.google.com/drive/folders/1XGKEwtNActy6Gf1coKV3wqZeebFuV938?usp=sharing

STEP 3: DATA-PREPROCESSING

In [7]:

```python
from scipy.signal import find_peaks

def extract_features(df):
    extracted_features = []
    processed_cols = [col for col in df.columns if col not in ['Frame', 'Class']]

    for col in processed_cols:
        grouped_data = df.groupby(df.index // 60)[col]  # Group by 60 frames per minute
        extracted_features.append(grouped_data.mean().rename(f'{col}_mean')) # MEAN
        extracted_features.append(grouped_data.std().rename(f'{col}_std')) #STD
        extracted_features.append(grouped_data.min().rename(f'{col}_min')) #MIN
        extracted_features.append(grouped_data.max().rename(f'{col}_max')) # MAX
        extracted_features.append(grouped_data.apply(np.trapz).rename(f'{col}_auc')) # AUC
        extracted_features.append(grouped_data.apply(lambda x: len(find_peaks(x)[0])).rename(f'{col}_peaks')) #PEAK

    features_df = pd.concat(extracted_features, axis=1)
    return features_df
```


```python
    for col in processed_cols:
        grouped_data = df.groupby(df.index // 60)[col]  # Group by 60 frames per minute
        extracted_features.append(grouped_data.mean().rename(f'{col}_mean')) # MEAN
        extracted_features.append(grouped_data.std().rename(f'{col}_std')) #STD
        extracted_features.append(grouped_data.min().rename(f'{col}_min')) #MIN
        extracted_features.append(grouped_data.max().rename(f'{col}_max')) # MAX
        extracted_features.append(grouped_data.apply(np.trapz).rename(f'{col}_auc')) # AUC
        extracted_features.append(grouped_data.apply(lambda x: len(find_peaks(x)[0])).rename(f'{col}_peaks')) #PEAK

    features_df = pd.concat(extracted_features, axis=1)
    return features_df
```


```python
# Compute features
output_features = extract_features(combined)

# Add class column back to features_df
output_features['Class'] = combined['Class'].groupby(combined.index // 60).first().values
```

In [8]:

```python
output_features.info()
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

In [8]:
```
output_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 1201 entries, 0 to 1200
Columns: 109 entries, L5 x_mean to Class
dtypes: float64(90), int64(19)
memory usage: 1.0 MB
```

In [9]:
```
output_features.head()
```

Out[9]:

	L5 x_mean	L5 x_std	L5 x_min	L5 x_max	L5 x_auc	L5 x_peaks	L5 y_mean	L5 y_std	L5 y_min	L5 y_max	...
0	0.011449	0.133862	-0.315158	0.375158	0.609679	16	-0.012295	0.121853	-0.276572	0.320082	...
1	-0.031521	0.351144	-0.805051	1.370930	-1.505659	16	0.011113	0.201366	-0.500589	0.404246	...
2	0.022407	0.605444	-1.303408	1.292384	1.593019	18	0.025555	0.431345	-0.877882	0.872093	...
3	0.035019	0.568312	-1.107645	1.375268	1.960137	13	-0.137832	0.480056	-1.120734	1.046369	...
4	-0.072219	0.699495	-1.870132	1.844777	-3.954333	14	0.027697	0.913177	-2.486366	3.146031	...

5 rows × 109 columns

6. Step 4: Training

a. Summary of accuracy table with boning and slicing meat data using SVM model

SVM model	Train-test split	Cross-validation
Original features	84.49%	84.29%
With hyperparameter tuning	85.32%	85.71%
With feature selection and hyperparameter tuning	82.83%	82.86%

Student name: Nguyen Anh Tuan
Student ID: 103805949

With PCA and hyper parameter tuning	83.38%	82.02%

[10]:

```
from sklearn.model_selection import train_test_split, cross_v
al_score, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.feature_selection import SelectKBest, f_classif
```

In [11]:

```
X = output_features.drop(columns=["Class"], axis=1)
y = output_features['Class'] # Target variable

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random
_state=42)

svm = SVC()
svm.fit(X_train, y_train)
result = svm.score(X_test, y_test)
print(f"Train test split accuracy: {result * 100:.2f}%")
```

```
Train test split accuracy: 84.49%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

2) 10-fold cross validation

In [12]:
```
scores = cross_val_score(svm, X_train, y_train, cv=10)
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 100:.2f}%")
```

```
10-fold cross validation accuracy: 84.29%
```

In [13]:
```
from sklearn.model_selection import GridSearchCV
# defining parameter range
param_grid = {'C': [0.1, 1, 10],
              'kernel': ['linear', 'rbf']}
grid = GridSearchCV(SVC(), param_grid, cv=10)
# fitting the model for grid search
grid.fit(X_train, y_train)
```

Out[13]:
```
▸ GridSearchCV
▸ estimator: SVC
    ▸ SVC
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

In [14]:
```
optimal_svm = grid.best_estimator_
```

In [15]:
```
result = optimal_svm.score(X_test, y_test)
print(f"Train test split accuracy: : {result * 100:.2f}%")
```

```
Train test split accuracy: : 85.32%
```

In [16]:
```
scores = cross_val_score(optimal_svm, X_train, y_train, cv=1
0, scoring='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 10
0:.2f}%")
```

```
10-fold cross validation accuracy: 85.71%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

4) 1 and 2 with hyper parameter tuning and 10 best features

In [17]:

```
X_new = SelectKBest(f_classif, k=10).fit_transform(X_train, y_train)

selector = SelectKBest(score_func=f_classif, k=10)
X_new_train = selector.fit_transform(X_train, y_train)
X_new_test = selector.transform(X_test)
optimal_svm.fit(X_new, y_train)

result = optimal_svm.score(X_new_test, y_test)
print(f"10 best features accuracy: {result * 100:.2f}%")
```

```
10 best features accuracy: 82.83%
```

In [18]:

```
scores = cross_val_score(optimal_svm, X_new_train, y_train, cv=10, scoring='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 100:.2f}%")
```

```
10-fold cross validation accuracy: 82.86%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

5) 1 and 2 with hyper parameter tuning and 10 principal components

In [19]:

```
new_pca = PCA(n_components=10)
X_train_pca = new_pca.fit_transform(X_train)
X_test_pca = new_pca.transform(X_test)
optimal_svm.fit(X_train_pca, y_train)
result = optimal_svm.score(X_test_pca, y_test)
print(f"PCA accuracy: {result * 100:.2f}%")
```

```
PCA accuracy: 83.38%
```

In [20]:

```
# 10-fold class validation with hyperparameter tuning
scores = cross_val_score(optimal_svm, X_train_pca, y_train, cv=10, scoring='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 100:.2f}%")
```

```
10-fold cross validation accuracy: 82.02%
```

b. Summary of accuracy table with boning and slicing meat data using different models

Model	Train-test split	Cross-validation
SGD	83.93%	79.88%
RandomForest	85.32%	83.69%

Student name: Nguyen Anh Tuan
Student ID: 103805949

MLP	83.38%	83.45%
SVM	84.49%	84.29%

Train SGD

In [21]:

```
# SGDclassifier
sgd = SGDClassifier()
sgd.fit(X_train, y_train)

result = sgd.score(X_test, y_test)
print(f"Train test split accuracy: : {result * 100:.2f}%")
```

```
Train test split accuracy: : 83.93%
```

In [22]:

```
scores = cross_val_score(sgd, X_train, y_train, cv=10, scorin
g='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 10
0:.2f}%")
```

```
10-fold cross validation accuracy: 79.88%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

Train RandomForest

In [23]:

```
# RandomForest

rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)

result = rf_classifier.score(X_test, y_test)
print(f"Train test split accuracy: : {result * 100:.2f}%")
```

```
Train test split accuracy: : 85.32%
```

In [24]:

```
scores = cross_val_score(rf_classifier, X_train, y_train, cv=
10, scoring='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 10
0:.2f}%")
```

```
10-fold cross validation accuracy: 83.69%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

Train MLP classifier

In [25]:

```
mlp = MLPClassifier(max_iter=500, random_state=42)
mlp.fit(X_train, y_train)
result = mlp.score(X_test, y_test)
print(f"Train test split accuracy: : {result * 100:.2f}%")
```

```
Train test split accuracy: : 83.38%
```

In [26]:

```
scores = cross_val_score(mlp, X_train, y_train, cv=10, scoring='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 100:.2f}%")
```

```
10-fold cross validation accuracy: 83.45%
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

In [26]:
```
scores = cross_val_score(mlp, X_train, y_train, cv=10, scorin
g='accuracy')
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 10
0:.2f}%")
```

```
10-fold cross validation accuracy: 83.45%
```

Train Original SVM

In [27]:
```
origin_svm = SVC()

origin_svm.fit(X_train, y_train)

result = origin_svm.score(X_test, y_test)
print(f"Train test split accuracy: {result * 100:.2f}%")
```

```
Train test split accuracy: 84.49%
```

In [28]:
```
scores = cross_val_score(svm, X_train, y_train, cv=10)
scores_mean = scores.mean()
print(f"10-fold cross validation accuracy: {scores_mean * 10
0:.2f}%")
```

```
10-fold cross validation accuracy: 84.29%
```

7. Step 5: Model Selection

- The best SVM model for my L5 and T12 datasets is the one with hyperparameter tuning. It achieved 85.32% on the train-test split, providing the highest accuracy, and demonstrating the best performance across different configurations.

- The RandomForest model with a train-test split (85.32%) was selected because it provides the highest accuracy and outperforms other models in this configuration.

8. Appendix

- Link to the code: https://www.kaggle.com/code/twananguyen/submit-portfolio-3
- Dataset: https://drive.google.com/drive/folders/1XGKEwtNActy6Gf1coKV3wqZeebFuV938?usp=sharing