Final Project

EECS 3401 Section A

Professor: Dr. Ruba Al Omari

Students: Andrew Hocking, Ranbir Khaira, Dexter King

1. Introduction

Physical fitness is a crucial aspect of maintaining a healthy lifestyle, and with the abundance of exercise options available, it becomes essential to identify effective and targeted workout routines. In this report, we delve into a dataset containing information about various fitness exercises, exploring the characteristics of each exercise, and evaluating the performance of machine learning algorithms in predicting exercise difficulty levels.

2. Dataset Overview

Our dataset comprised of 2918 different exercises, with the following nine attributes:

Column Name	**Column Description**	**Number of Missing Values in Column**
#	ID column (numeric from 0 to 2917)	0
Title	Title for the exercise (nominal)	0
Desc	Short description of the exercise (nominal)	1550
Type	Type of exercise (nominal: "Strength", "Stretching", "Cardio", etc.)	0
BodyPart	BodyPart that it targets (nominal: "Abdominals", "Quadriceps", "Biceps", etc.)	0
Equipment	Equipment needed for the workout (nominal: "Barbell", "Dumbbell", "Cable", etc.)	32
Level	Level of exercise (nominal: "Beginner","Intermediate", or "Expert")	0
Rating	Rating of exercise (numeric: 0 to 10)	1887
RatingDesc	Description for the rating (nominal: "Average)	2056

3. Framing the Problem

1. Supervised learning is being done, as the training examples are labeled.
2. The predictions being done are classification predictions, as each exercise will be classified into different categories (skill level).

3. Batch learning will be used since there is a large data set, no continuous flow of data being added to it, and therefore no need to adjust to changing data.

4. Looking at the Big Picture

The goal is to determine the skill level of an exercise based on features such as the equipment, targeted body part, and type of exercise. This will be useful for people who would like to try different exercises but do not know if it is practical for them to do it given their skill sets.

5. Exploratory Data Analysis (EDA)

5.1. Exercise Distribution by Type

87% of all exercises in the dataset are strength exercises, which are then followed by stretching exercises and plyometric exercises (see Figure 1). This is important as it is an indicator that this analysis will be more useful for those interested in the skill levels of strength exercises rather than other types, as most of the machine learning training will be based on strength exercises.

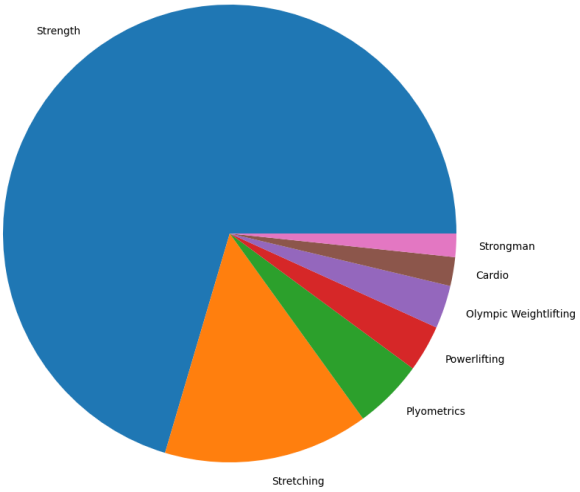


Figure 1

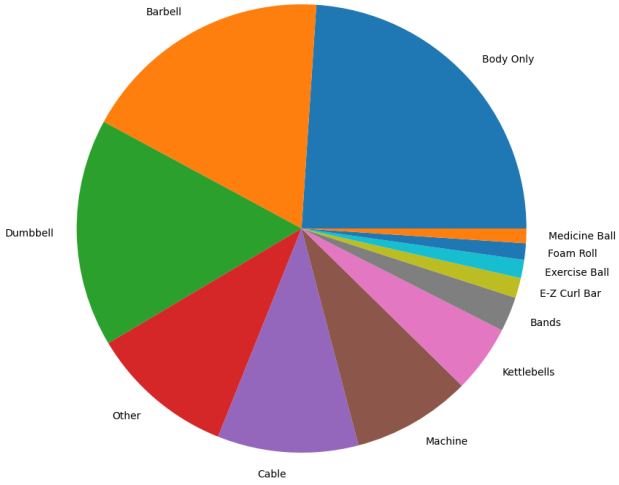


Figure 2

5.2. Equipment Usage

Another useful detail for those who will use this analysis in the future, is the fact that the majority of exercises in the dataset require no equipment or very basic equipment. This means that the analysis will be most accurate for finding the skill level of exercises that require no equipment (body-weight/calisthenics) as 24% of the dataset falls into this category (see Figure 2). For equipment, dumbbells and barbells are the most common,

which is useful information for those who are looking for the skill-levels of exercises which use that equipment.

5.3. Exercise Distribution by Skill Level

Perhaps most interesting for this application, is how unevenly distributed the skill levels were in this dataset. With a total of 2918 exercises in the dataset, a whopping 2446 of them were labelled as “Intermediate”-level, or 83.8% (see Figure 3). This was followed by 459 “Beginner”-level exercises, about 15.7%. Which means that out of over 2900 exercises, only a measly 13 of them were labelled as “Expert”-level – only 0.5%.

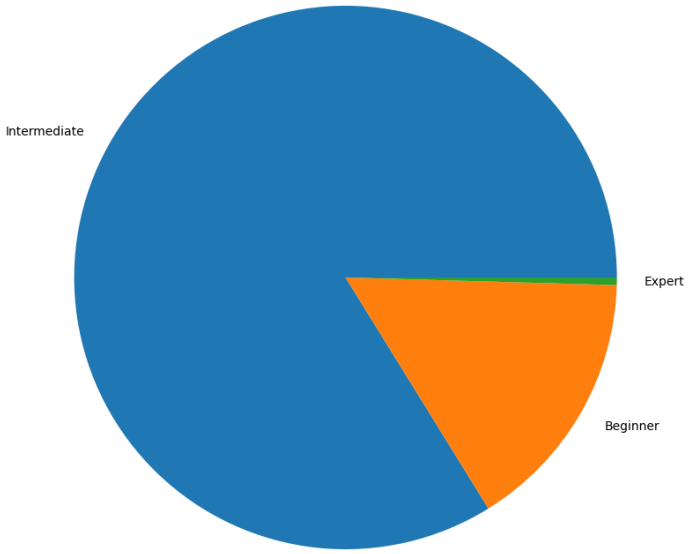


Figure 3

6. Data Cleaning

There are no duplicate values in the dataset, so that issue was ignored. There were however 1550 missing values for description, 1887 missing values for rating, and 2056 missing values for rating description. Because the description and rating descriptions are irrelevant for classifying the data into skill levels, the attributes were deleted altogether. Entries with missing ratings were also deleted altogether. To help with classification, all skill level attributes were changed into numbers, with ‘beginner’ being changed to 0, ‘intermediate’ to 1, and ‘expert’ to 2. Next, using a pipeline, the numerical columns were scaled, and the categorical columns were encoded to prepare for machine learning. An imputer was not used since we had no more missing values. This pipeline transformation resulted in 820 features.

7. Machine Learning Models

Since we are classifying exercises into different skill levels, we will be using classification models. We trained three machine learning models – Support Vector Machine (SVM), K Nearest Neighbors (KNN), and Decision Tree – to predict exercise difficulty levels. We used SVM since it is good when data is not regularly distributed, like ours. KNN is accurate for when data is significantly larger than the number of features, which is the case. Finally, the decision tree was used as it is also good for irregularly distributed data, and furthermore good for categorical data, which is what most of our features are.

7.1. Model Performance

The mean squared error (MSE) was used as the evaluation metric for its simplicity. The data was split into a 80% training dataset.

1. **Support Vector Machine (SVM):** Achieved an MSE of 0.3481, indicating a relatively accurate prediction of exercise difficulty levels.
2. **K Nearest Neighbors (KNN):** Produced an MSE of 0.3544, suggesting a comparable performance to SVM.
3. **Decision Tree:** The Decision Tree model yielded an MSE of 0.3734, indicating slightly lower predictive accuracy compared to SVM and KNN.

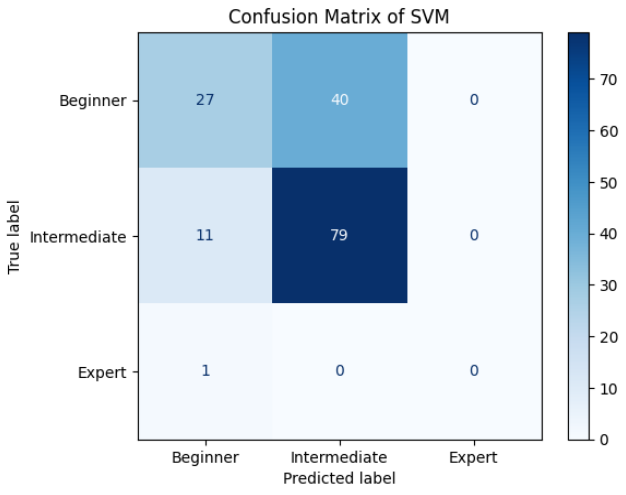


Figure 4 - Confusion matrix comparing predicted difficulty levels to actual difficulty levels using SVM.

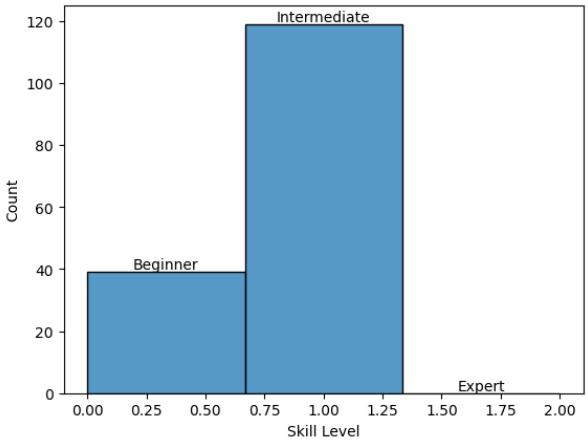


Figure 5 - Histogram showing the distribution of predicted difficulty levels using SVM.

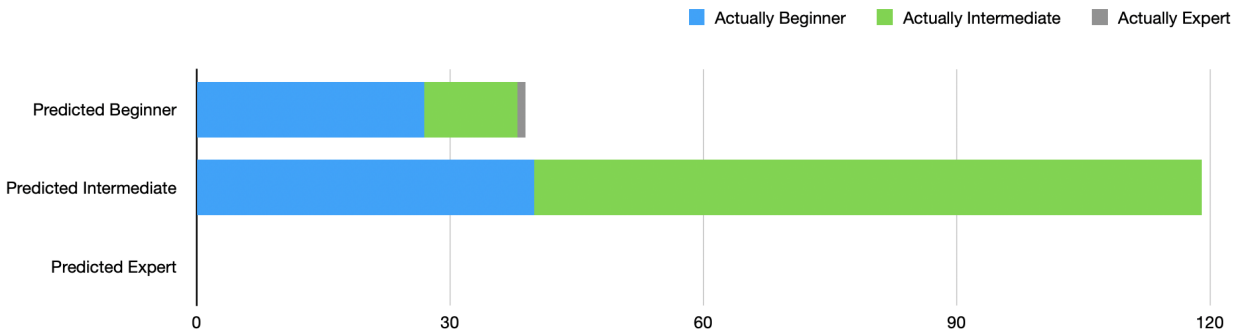


Figure 6 - Stacked bar graph comparing predicted difficulty levels to actual difficulty levels using SVM.

7.2. Model Analysis

The SVM model outperformed KNN and Decision Tree in terms of MSE, suggesting it may be the most effective model for predicting exercise difficulty levels in this dataset. However, further hyperparameter tuning and feature engineering could enhance the performance of all models. For example, in this case, all three algorithms failed to accurately identify the “Expert” level exercise that was in the training set (see Figures 3 through 6). With a larger training set that included more “Expert” level exercises, we could have likely avoided this outcome.

8. Limitations

1. **Missing Data:** The data was simplified because of the amount of missing values. Two attributes were removed, along with many entries that had missing ratings. If these values were not missing, it could have led to a more accurate prediction.
2. **Model Complexity:** The machine learning models used did quite well, but are still relatively simple, the exploration of complex and more thorough machine learning models, may have yielded better performance.
3. **Small Dataset:** While this dataset was not unusably small, with just under 3000 entries, a much larger dataset would allow for a larger training set, likely resulting in a more accurate model.

9. Conclusion

This analysis provides valuable insights into the fitness exercise dataset, highlighting trends in exercise types, body part targeting, and equipment usage. The machine learning models show promising results, with SVM exhibiting the best performance in predicting exercise difficulty levels. However, addressing missing data, obtaining a

larger dataset, and exploring more sophisticated models could further enhance the analysis and prediction accuracy.

10. Appendix 1

November 21, 2023

1 Gym Excercise Dataset Description

The description is taken directly from the owner Niharika Pandit

Context

This is a dataset created for analyzing and evaluating workouts that one can do at the gym(or at home) to stay healthy. Exercising and being fit is becoming very important and almost a daily routine for all individuals and what better than to take a data-driven route for success to meet one's fitness goals.

Inspiration

If you go to a gym, the first thing you realize is the myriad of exercises available to do. The exercises range from bodyweight, machine-based or dumbbell/barbell based. With so many exercises to do, beginners or even professional can wonder which the exercise that will target a specific muscle the best and that is where this analysis can be useful. I also thought it would be fun to visualize the exercise details.

Content

There is one file with 9 columns for each exercise. Columns may contain null values as the data is raw and scraped from various internet sources.

Attributes for megaGymDataset.csv:

1. '#' - ID column (numeric from 0 to 2917)
2. Title - Title for the excercise (nominal)
3. Desc - Short description of the excercise (nominal)
4. Type - Type of excercise (nominal: "Strength", "Stretching", "Cardio", etc.)
5. BodyPart - BodyPart that it targets (nominal: "Abdominals", "Quadriceps", "Biceps", etc.)
6. Equipment - Equipment needed for the workout (nominal: "Barbell", "Dumbbell", "Cable", etc.)
7. Level - Level of excercise (nominal: "Beginner","Intermediate", or "Expert")
8. Rating - Rating of excercise (numeric: 0 to 10)
9. RatingDesc - Description for the rating (nominal: "Average)

2 1. Look at the big picture and frame the problem.

2.1 Frame the problem

1. Supervised learning - training examples are labeled.

2. Classification - Predict the skill level of the exercise.
3. Batch learning

- Large data set
- No continuous flow of data coming into the system
- No need to adjust to changing data rapidly

2.2 Look at the big picture

Determine the skill level of the exercise based on features such as the equipment, targeted body part, and type of exercise.

[2]:
```
# Import libraries

import sklearn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
```

3 2. Load the dataset and examine the data structure

[3]:
```
url = "https://raw.githubusercontent.com/AndrewHocking/EECS-3401-Final-Project/
 ↪main/megaGymDataset.csv"
gym = pd.read_csv(url, sep=',')

#Fix ID column name
gym.rename(columns={"Unnamed: 0": "ID"}, inplace = True)

gym_backup = gym

gym.head()
```

[3]:

	ID	Title	Desc	Type	BodyPart
0	0	Partner plank band row	The partner plank band row is an abdominal exe…	Strength	Abdominals
1	1	Banded crunch isometric hold	The banded crunch isometric hold is an exercis…	Strength	Abdominals
2	2	FYR Banded Plank Jack	The banded plank jack is a variation on the pl…	Strength	Abdominals
3	3	Banded crunch	The banded crunch is an exercise targeting the…	Strength	Abdominals
4	4	Crunch	The crunch is a popular core exercise targetin…	Strength	Abdominals

```
  Equipment         Level  Rating RatingDesc
0     Bands  Intermediate     0.0        NaN
1     Bands  Intermediate     NaN        NaN
2     Bands  Intermediate     NaN        NaN
3     Bands  Intermediate     NaN        NaN
4     Bands  Intermediate     NaN        NaN
```

Average rating for all excercises is 5.9. The bottom 25% of excercises are rated 3 or lower. The top 25% of excercises are rated 8.7 or higher.

[4]:
```
gym.describe()
```

[4]:
```
                ID       Rating
count  2918.000000  1031.000000
mean   1458.500000     5.919690
std     842.498368     3.584607
min       0.000000     0.000000
25%     729.250000     3.000000
50%    1458.500000     7.900000
75%    2187.750000     8.700000
max    2917.000000     9.600000
```

Missing values:

- Desc: 1550
- Rating: 1887
- RatingDesc: 2056

[5]:
```
gym.isnull().sum()
```

[5]:
```
ID              0
Title           0
Desc         1550
Type            0
BodyPart        0
Equipment      32
Level           0
Rating       1887
RatingDesc   2056
dtype: int64
```

[6]:
```
gym.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2918 entries, 0 to 2917
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   ID          2918 non-null   int64
 1   Title       2918 non-null   object
```

```
 2   Desc        1368 non-null   object
 3   Type        2918 non-null   object
 4   BodyPart    2918 non-null   object
 5   Equipment   2886 non-null   object
 6   Level       2918 non-null   object
 7   Rating      1031 non-null   float64
 8   RatingDesc  862 non-null    object
dtypes: float64(1), int64(1), object(7)
memory usage: 205.3+ KB
```

3.0.1 Excercise count for each type of excercise

[7]:
```
gym['Type'].value_counts()
```

[7]:
```
Type
Strength                2545
Stretching               147
Plyometrics               97
Powerlifting              37
Cardio                    35
Olympic Weightlifting     35
Strongman                 22
Name: count, dtype: int64
```

3.0.2 Excercise count for each body part

[8]:
```
gym['BodyPart'].value_counts()
```

[8]:
```
BodyPart
Abdominals     662
Quadriceps     646
Shoulders      340
Chest          262
Biceps         168
Triceps        151
Lats           124
Hamstrings     121
Middle Back    118
Lower Back      97
Glutes          81
Calves          47
Forearms        31
Traps           24
Abductors       21
Adductors       17
Neck             8
Name: count, dtype: int64
```

3.0.3 Excercise count for each type of equipment

[9]:
```
gym['Equipment'].value_counts()
```

[9]:
```
Equipment
Body Only        1078
Dumbbell          516
Barbell           282
Other             254
Cable             226
Machine           175
Kettlebells       149
Bands             100
Medicine Ball      38
Exercise Ball      35
E-Z Curl Bar       22
Foam Roll          11
Name: count, dtype: int64
```

3.0.4 Excercise count for each level

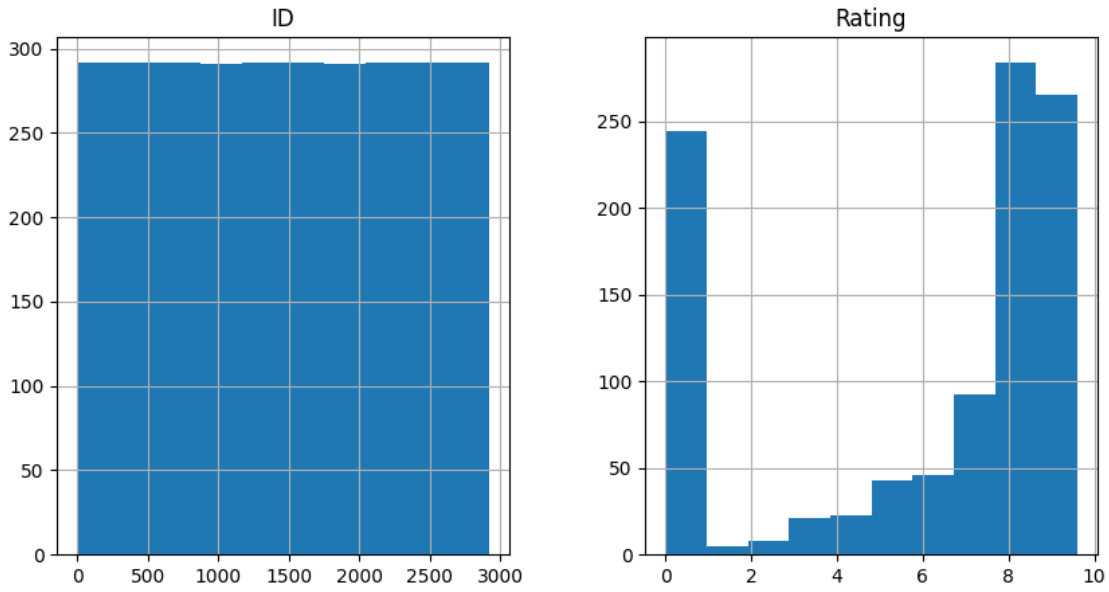
[10]:
```
gym['Level'].value_counts()
```

[10]:
```
Level
Intermediate    2446
Beginner         459
Expert            13
Name: count, dtype: int64
```

4 3. Exploratory Data Analysis

There appears to be a large number of excercises with a zero rating which suggests that these excercises are likely not rated correctly.

[11]:
```
gym.hist(figsize=(10, 5))
plt.show()
```

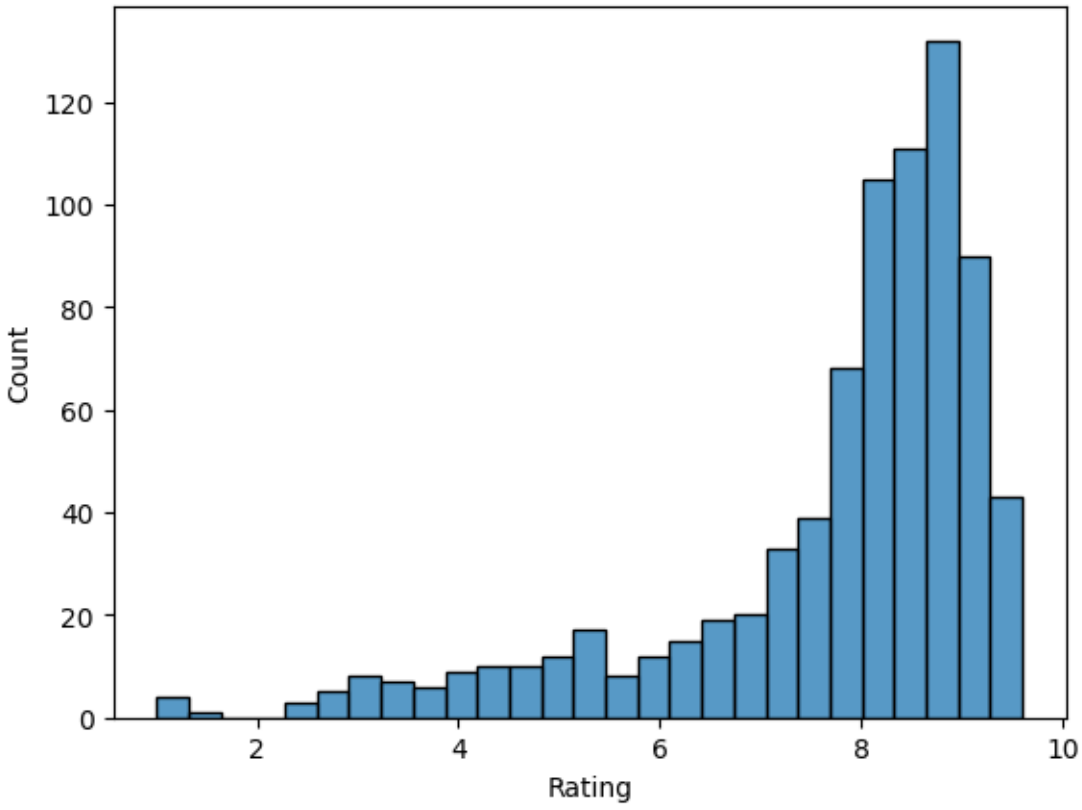


4.0.1 Distribution of ratings for all excercises

After excluding excercises with a 0 rating, a large amount of the excercises seem to fall between the 8 to 10 range suggesting that most excercises are effective at targetting their specific muscle groups.

[12]:
```
#Create rating distribution excluding all the excercises rated 0
gym_exclude = gym[gym['Rating']!= 0.0]
sns.histplot(gym_exclude, x='Rating')
```

[12]:
```
<Axes: xlabel='Rating', ylabel='Count'>
```



4.0.2 Distribution of excercises for each muscle group.

Abdominals and Quadriceps have the highest number of excercises followed by the rest of the major muscle groups that are most often targeted. Unsurprisingly, neck, adductors, abductors, traps and forearms have lowest number of excercises as they are not often trained directly.

[13]:
```
#Create distribution of excercises for each muscle group
bodypart_graph = gym_exclude.groupby(['BodyPart']).count().sort_values(by='ID')
px.bar(bodypart_graph, x = bodypart_graph.index, y='ID')
```

4.0.3 Boxplot of excercise rating for each equipment

This graph shows which types of equipment tend to have excercises of higher ratings. This information, combined with the total number of excercises associated with each type of equipment can be used to determine which equipment make the best investment.

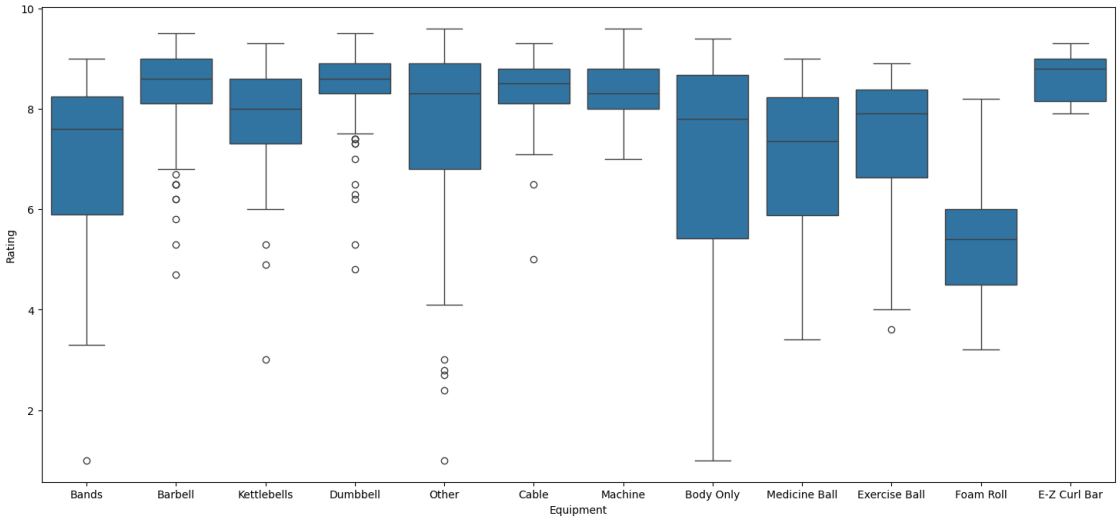
Looking at the data, it appears that dumbbells and barbells are the best equipment to invest in because of their relatively high average excercise ratings and the high number of excercises that require them. Even though the E-Z curl bar has the highest median, it also has the second least number of excercises.

Body only excercises are noteworthy as, even though the average rating is not high, the large amount of excercises combined with the large spread of excercise rating suggest that you can stil

get a good workout even as someone with no equipment provided a good selection of excercises.

[14]:
```
plt.figure(figsize=(18,8))
sns.boxplot(x='Equipment', y='Rating', data=gym_exclude)
gym['Equipment'].value_counts()
```

[14]:
```
Equipment
Body Only        1078
Dumbbell          516
Barbell           282
Other             254
Cable             226
Machine           175
Kettlebells       149
Bands             100
Medicine Ball      38
Exercise Ball      35
E-Z Curl Bar       22
Foam Roll          11
Name: count, dtype: int64
```



4.0.4 Distribution of body only excercises for each muscle group

From this graph, it appears that a majority of body weight excercises target the quadriceps and abdominals. However, there appears to be a decent amount of excercises for most of the other major muscle groups that tend to be targeted directly. However, there appears to be little bodyweight only excercise variety for biceps and calves.

[15]:
```
#Create distribution of bodyweight only excercises for each muscle group
bodyonly_graph = gym_exclude[gym_exclude.Equipment == 'Body Only']
```

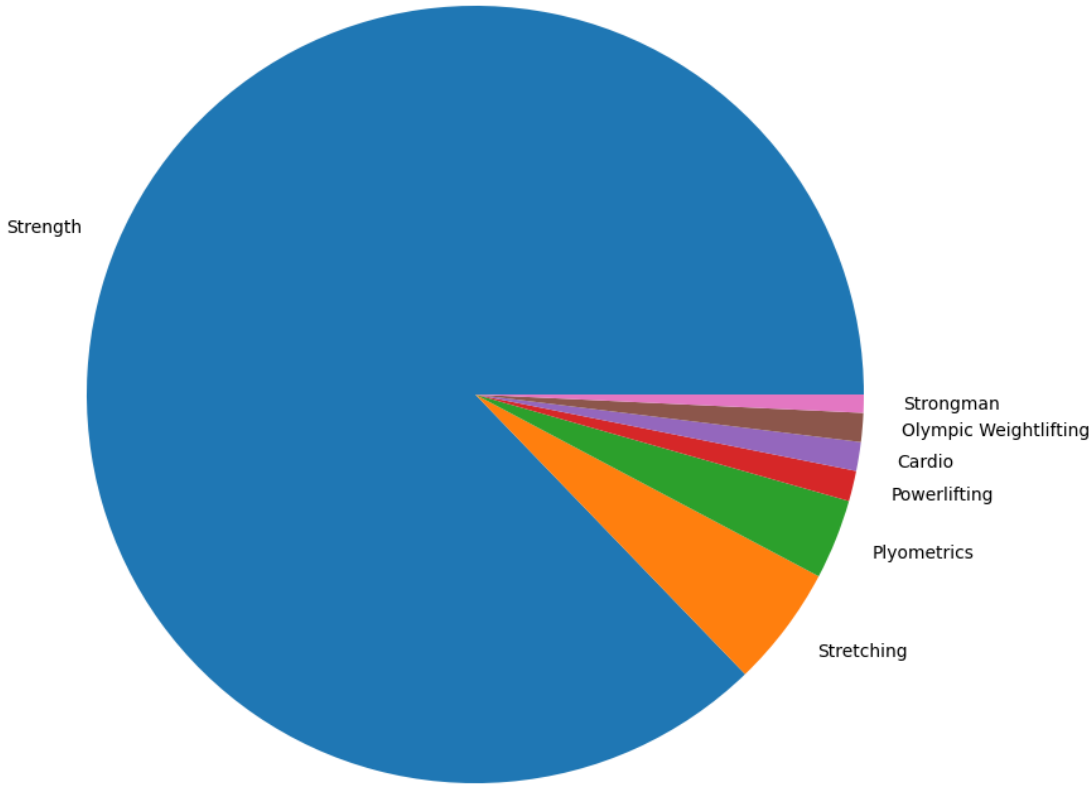
```
bodyonly_graph = bodyonly_graph.groupby(['BodyPart']).count().
  ↪sort_values(by='ID')
px.bar(bodyonly_graph, x = bodyonly_graph.index, y='ID')
```

4.0.5 Exercise Distribution by Type

87% of all exercises in the dataset are strength exercises, which are then followed by stretching exercises and plyometric exercises. This is important as it is an indicator that this analysis will be more useful for those interested in the skill levels of strength exercises rather than other types, as most of the machine learning training will be based on strength exercises.

[16]:
```
gym["Type"].value_counts().plot.pie(figsize=(10, 10), ylabel="")
gym['Type'].value_counts()
```

[16]:
```
Type
Strength                2545
Stretching               147
Plyometrics               97
Powerlifting              37
Cardio                    35
Olympic Weightlifting     35
Strongman                 22
Name: count, dtype: int64
```



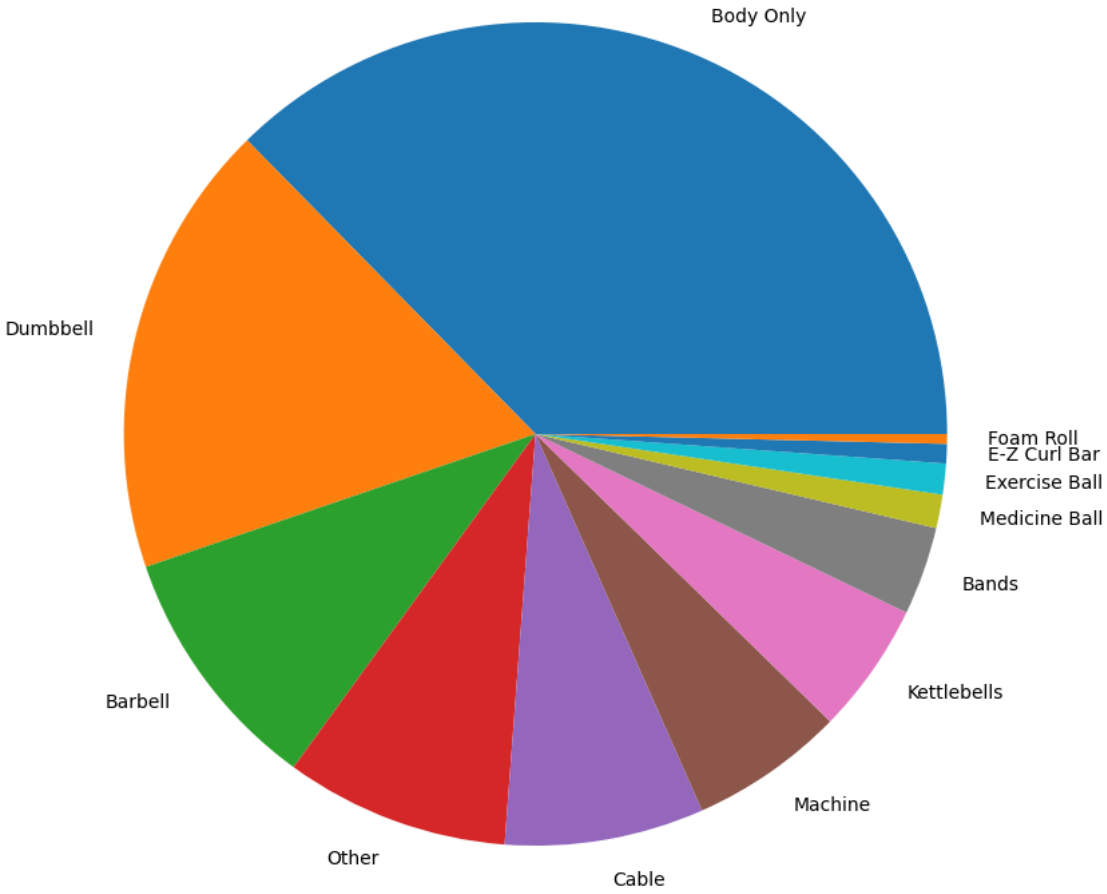
4.0.6 Exercise Distribution by Equipment Usage

Another useful detail for those who will use this analysis in the future, is the fact that the majority of exercises in the dataset require no equipment or very basic equipment. This means that the analysis will be most accurate for finding the skill level of exercises that require no equipment (body-weight/calisthenics) as 24% of the dataset falls into this category. For equipment, dumbbells and barbells are the most common, which is useful information for those who are looking for the skill-levels of exercises which use that equipment.

[17]:
```
gym["Equipment"].value_counts().plot.pie(figsize=(10, 10), ylabel="")
gym['Equipment'].value_counts()
```

[17]:
```
Equipment
Body Only        1078
Dumbbell          516
Barbell           282
Other             254
```

```
Cable             226
Machine           175
Kettlebells       149
Bands             100
Medicine Ball      38
Exercise Ball      35
E-Z Curl Bar       22
Foam Roll          11
Name: count, dtype: int64
```



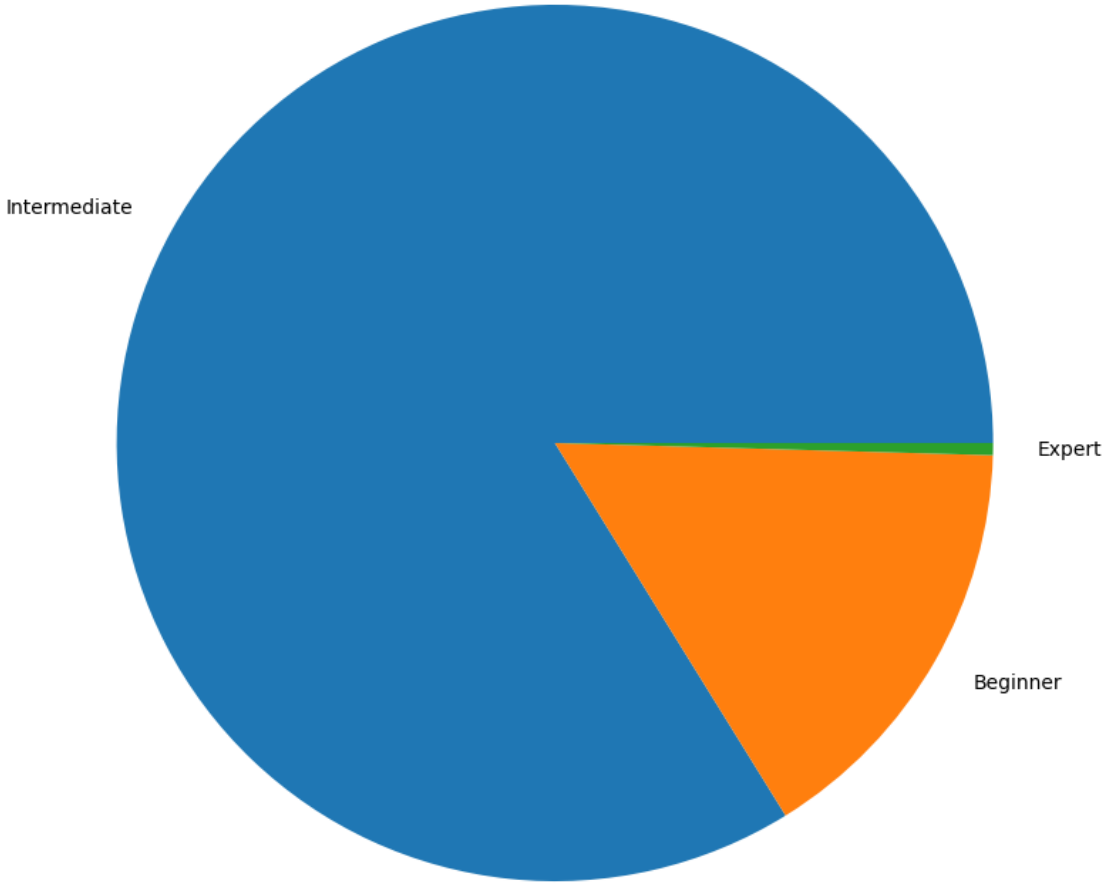
4.0.7 Exercise Distribution by Skill Level

Perhaps most interesting for this application, is how unevenly distributed the skill levels were in this dataset. With a total of 2918 exercises in the dataset, a whopping 2446 of them were labelled as “Intermediate”-level, or 83.8%. This was followed by 459 “Beginner”-level exercises, about

15.7%. Which means that out of over 2900 exercises, only a measly 13 of them were labelled as "Expert"-level – only 0.5%.

[18]:
```
gym["Level"].value_counts().plot.pie(figsize=(10, 10), ylabel="")
gym['Level'].value_counts()
```

[18]:
```
Level
Intermediate    2446
Beginner         459
Expert            13
Name: count, dtype: int64
```



5 4. Prepare the data for Machine Learning Algorithms

There are no duplicate entries

[19]:
```
#Check for duplicates
gym.duplicated().sum()
```

[19]:
```
0
```

There are missing values for Desc, Rating and RatingDesc. However, Desc and RatingDesc are irrelevant for the algorithm so they will be deleted.

[20]:
```
#Find the number of missing values in each column

gym.isna().sum()
```

[20]:
```
ID              0
Title           0
Desc         1550
Type            0
BodyPart        0
Equipment      32
Level           0
Rating       1887
RatingDesc   2056
dtype: int64
```

Drop irrelevant columns Desc and RatingDesc

[21]:
```
#Drop columns Desc and RatingDesc
gym.drop(labels=['Desc'], axis=1, inplace=True)
gym.drop(labels=['RatingDesc'], axis=1, inplace=True)

gym.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2918 entries, 0 to 2917
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   ID         2918 non-null   int64
 1   Title      2918 non-null   object
 2   Type       2918 non-null   object
 3   BodyPart   2918 non-null   object
 4   Equipment  2886 non-null   object
 5   Level      2918 non-null   object
 6   Rating     1031 non-null   float64
dtypes: float64(1), int64(1), object(5)
memory usage: 159.7+ KB
```

Remove all entries with missing ratings

[22]:
```
#Change 0.0 ratings to NaN
gym = gym.replace(0.0,np.nan)

gym.dropna(subset=["Rating"], inplace=True)

gym.isna().sum()
```

[22]:
```
ID            0
Title         0
Type          0
BodyPart      0
Equipment    27
Level         0
Rating        0
dtype: int64
```

Convert all exercise levels to numbers: - 0 = Beginner - 1 = Intermediate - 2 = Expert

[23]:
```
# Number of initial values
gym['Level'].value_counts()
```

[23]:
```
Level
Intermediate    463
Beginner        317
Expert            7
Name: count, dtype: int64
```

[24]:
```
gym = gym.replace("Beginner",0)
gym = gym.replace("Intermediate",1)
gym = gym.replace("Expert",2)

#Number of replaced values is the same
gym['Level'].value_counts()
```

[24]:
```
Level
1    463
0    317
2      7
Name: count, dtype: int64
```

Create pipeline that will scale numerical columns using StandardScaler and encode categorical columns using OneHotEncoder.

[25]:
```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
```

[26]:
```
# Create the cat and num columns

# Get a list of column names from the 'gym' DataFrame that are of numerical␣
 ↪data types.
num_cols = gym.select_dtypes(include='number').columns.to_list()
# Get a list of column names from the 'gym' DataFrame that are not of numerical␣
 ↪data types.
cat_cols = gym.select_dtypes(exclude='number').columns.to_list()


# Exclude the target from numerical columns
num_cols.remove("Level")

# Create pipelines for numeric and categorical columns, imputer not used since␣
 ↪no empty values.
num_pipeline = make_pipeline(StandardScaler())
cat_pipeline = make_pipeline(OneHotEncoder(sparse_output=False))

# Use ColumnTransformer to set the estimators and transformations

preprocessing = ColumnTransformer([('num', num_pipeline, num_cols),
                                   ('cat', cat_pipeline, cat_cols)],
                                  remainder='passthrough')
```

[27]:
```
preprocessing
```

[27]:
```
ColumnTransformer(remainder='passthrough',
                  transformers=[('num',
                                 Pipeline(steps=[('standardscaler',
                                                  StandardScaler())]),
                                 ['ID', 'Rating']),
                                ('cat',
                                 Pipeline(steps=[('onehotencoder',
OneHotEncoder(sparse_output=False))]),
                                 ['Title', 'Type', 'BodyPart', 'Equipment'])])
```

[28]:
```
gym_prepared = preprocessing.fit_transform(gym)

# Scikit-learn strips the column headers, so just add them back on afterward.
feature_names=preprocessing.get_feature_names_out()
gym_prepared = pd.DataFrame(data=gym_prepared, columns=feature_names)
```

[29]:
```
gym_prepared.columns
```

[29]:
```
Index(['num__ID', 'num__Rating', 'cat__Title_3/4 sit-up',
       'cat__Title_90-degree jump squat', 'cat__Title_90/90 Hamstring',
       'cat__Title_Ab Crunch Machine', 'cat__Title_Ab Roller',
```

```
       'cat__Title_Ab bicycle', 'cat__Title_Adductor SMR',
       'cat__Title_Adductor/Groin',
       …
       'cat__Equipment_Dumbbell', 'cat__Equipment_E-Z Curl Bar',
       'cat__Equipment_Exercise Ball', 'cat__Equipment_Foam Roll',
       'cat__Equipment_Kettlebells', 'cat__Equipment_Machine',
       'cat__Equipment_Medicine Ball', 'cat__Equipment_Other',
       'cat__Equipment_nan', 'remainder__Level'],
      dtype='object', length=821)
```

6 5. Train and evaluate different machine learning algorithms

Since we are predicting a category (skill level) we will be using classification models. We will use MSE as a performance metric, because of its simplicity. We will be training Support Vector Machine (SVM), K Nearest Neighbour (KNN), and Decision Tree models. SVM is a good model for data that is not regularly distributed, which our data is not. KNN is a solid model to use when the data is significantly larger than the number of features, which is the case here, given our 9 features being over 300 times smaller than our 2918 total exercises. Finally, the Decision Tree model is good for irregularly distributed data just like SVM, and it is good for categorical data, which is important, because most of our features are categorical.

Split the dataset into a training dataset (80%) and testing dataset.

[30]:
```
from sklearn.model_selection import train_test_split


X = gym_prepared.drop(["remainder__Level"], axis=1)
y = gym_prepared["remainder__Level"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
```

```
(629, 820) (629,) (158, 820) (158,)
```

Train SVM Model

[31]:
```
#import and train
from sklearn.svm import SVC

model_svm = SVC(kernel='poly', C=0.1, gamma=1)

model_svm.fit(X_train, y_train.values.ravel())
```

[31]:
```
SVC(C=0.1, gamma=1, kernel='poly')
```

Test your model on the test set, and report on the Mean Squared Error

[32]:
```
#import
from sklearn.metrics import mean_squared_error as mse

#make predictions
svm_y_predict = model_svm.predict(X_test)

svm_mse = mse(y_test, svm_y_predict)
print(f'SVM MSE: {svm_mse}')
```

```
SVM MSE: 0.34810126582278483
```

Train K Neighbours Model

[33]:
```
#import and train
from sklearn.neighbors import KNeighborsClassifier

model_knn = KNeighborsClassifier(3)

model_knn.fit(X_train, y_train.values.ravel())
```

[33]:
```
KNeighborsClassifier(n_neighbors=3)
```

Test your model on the test set, and report on the Mean Squared Error

[34]:
```
#make predictions
knn_y_predict = model_knn.predict(X_test)

knn_mse = mse(y_test, knn_y_predict)
print(f'K Neighbours MSE: {knn_mse}')
```

```
K Neighbours MSE: 0.35443037974683544
```

Train Decision Tree Model

[35]:
```
#import and train
from sklearn.tree import DecisionTreeClassifier

model_decisiontree = DecisionTreeClassifier(max_depth=10, random_state=42)

model_decisiontree.fit(X_train, y_train.values.ravel())
```

[35]:
```
DecisionTreeClassifier(max_depth=10, random_state=42)
```

Test your model on the test set, and report on the Mean Squared Error

[36]:
```
#make predictions
decision_tree_y_predict = model_decisiontree.predict(X_test)

decision_tree_mse = mse(y_test, decision_tree_y_predict)
print(f'Decision Tree MSE: {decision_tree_mse}')
```

```
Decision Tree MSE: 0.37341772151898733
```

Compare Results of all models:

[37]:
```
performance_data = {
    'Model': ['SVM', 'K Nearest Neighbour', 'Decision Tree'],
    'MSE': [svm_mse, knn_mse, decision_tree_mse]
}

performance_df = pd.DataFrame(performance_data)

# Display the performance comparison table
performance_df
```

[37]:

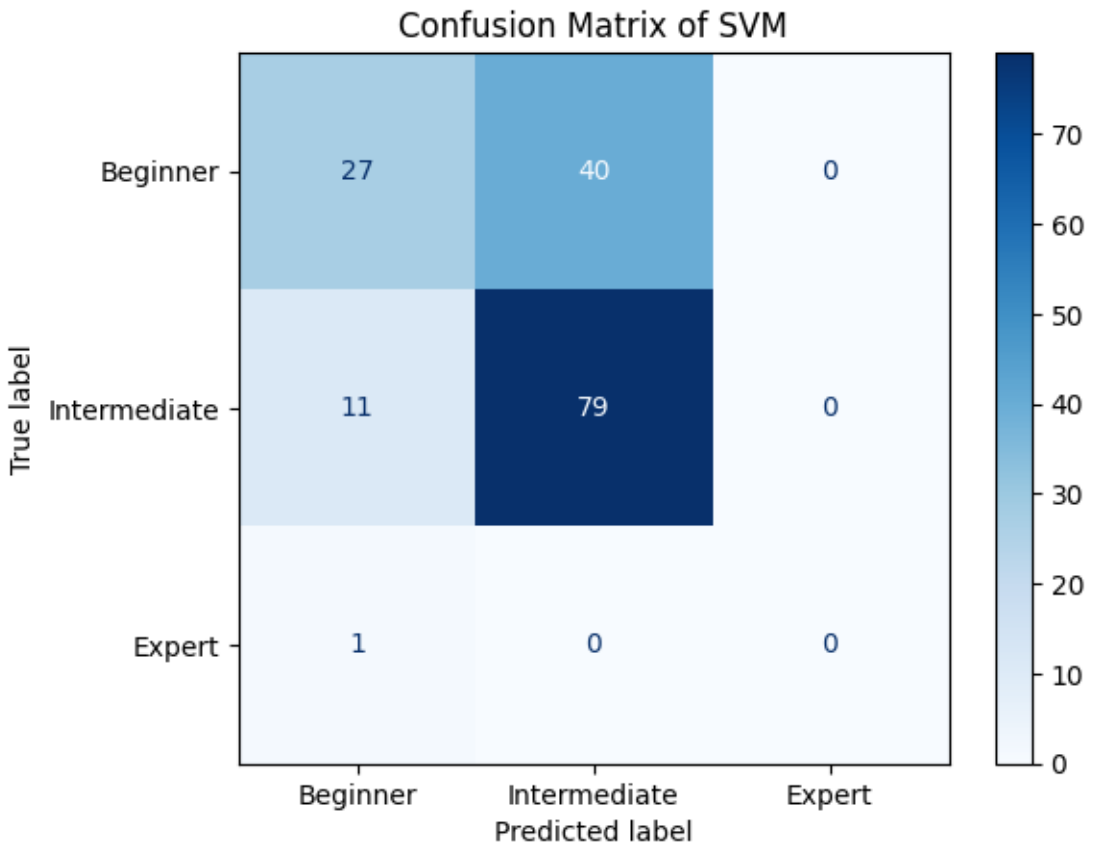
	Model	MSE
0	SVM	0.348101
1	K Nearest Neighbour	0.354430
2	Decision Tree	0.373418

Plot the prediction vs. actual for the best performing model (in this case, SVM).

[38]:
```
from sklearn.metrics import ConfusionMatrixDisplay

disp = ConfusionMatrixDisplay.from_estimator(
        model_svm,
        X_test,
        y_test,
        display_labels=["Beginner", "Intermediate", "Expert"],
        cmap=plt.cm.Blues,
)
disp.ax_.set_title("Confusion Matrix of SVM")

plt.show()
print("Actual values:")
y_test.replace(0, "Beginner").replace(1, "Intermediate").replace(2, "Expert").
 ↪value_counts()
```

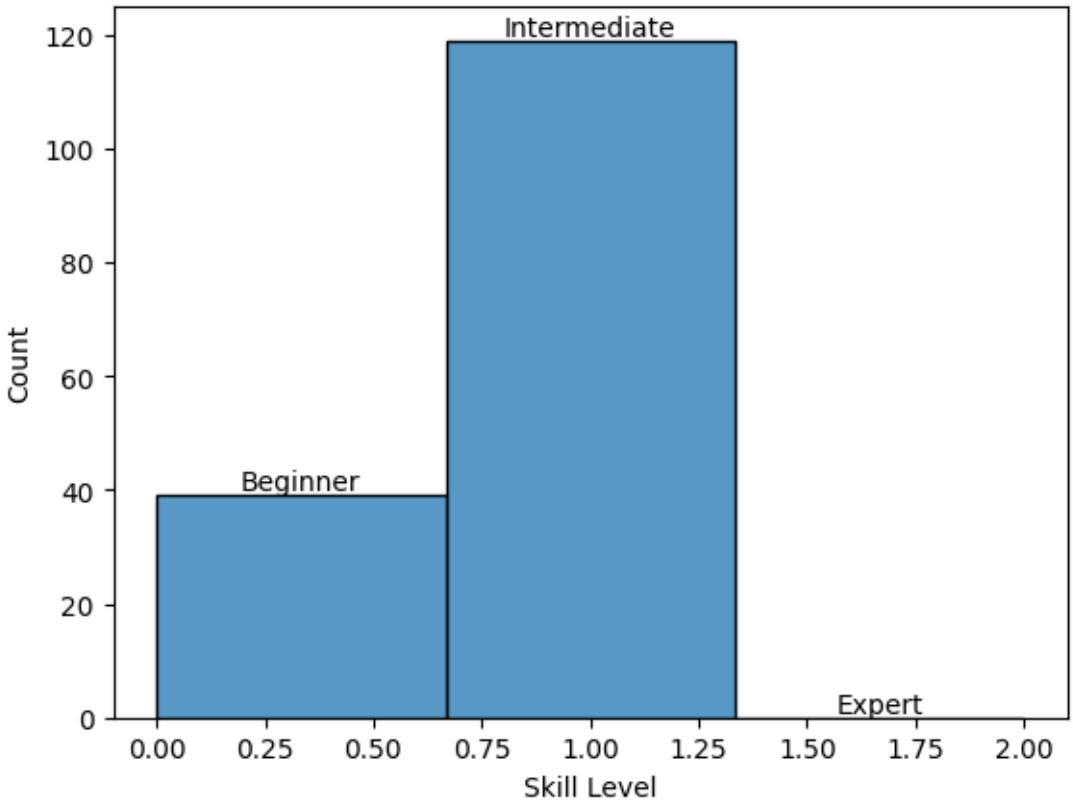



```
Actual values:
```

[38]:
```
remainder__Level
Intermediate    90
Beginner        67
Expert           1
Name: count, dtype: int64
```

Plot a histogram showing the distribution of predicted difficulty levels using SVM.

[39]:
```
labels = ["Beginner", "Intermediate", "Expert"]
graph = sns.histplot(svm_y_predict, bins=3, binrange=(0,2))
graph.set(xlabel="Skill Level")
graph.bar_label(graph.containers[0], labels)
plt.show()
```


Intermediate
Beginner
Expert
Count
Skill Level
120
100
80
60
40
20
0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

11. Appendix 2

Our video can be found here:
https://youtu.be/827EFCVQjys

The project source code and a copy of dataset can be found here:
https://github.com/AndrewHocking/EECS-3401-Final-Project

The original dataset can be found here:
https://www.kaggle.com/datasets/niharika41298/gym-exercise-data