IMPROVE GAME ANALYTICS USING ONLINE ML TO IDENTIFY PLAYER PROFILES IN SPACE WARS

EXISTING ISSUES:

- THE EXISTING APP DOES NOT UTILIZE ONLINE MACHINE LEARNING (ML) FOR USER BEHAVIOR ANALYTICS, INCLUDING BOTH CLASSIFICATION AND CLUSTERING.

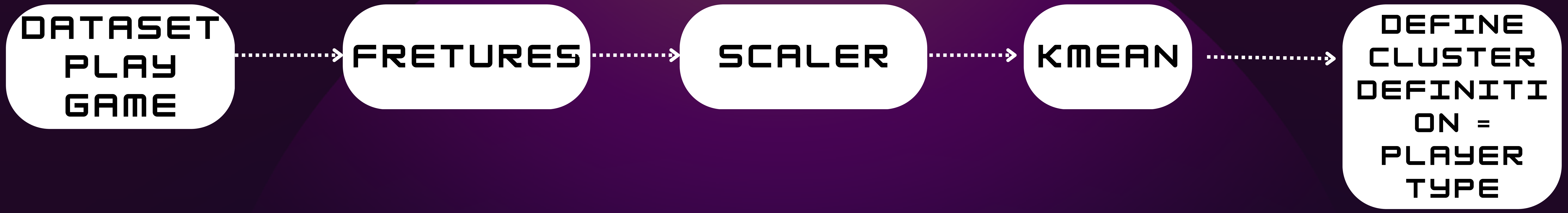
CURRENTLY, IT ONLY EMPLOYS OFFLINE MACHINE LEARNING.

OBJECTIVE:

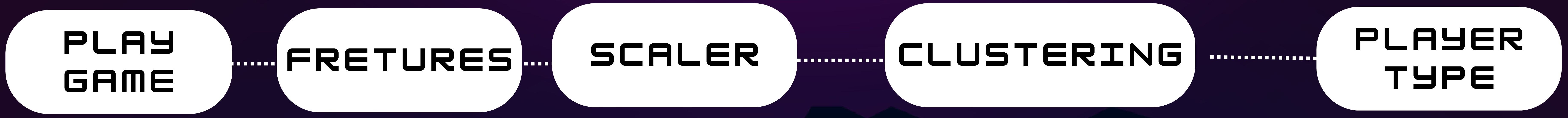
1. INTEGRATE ONLINE ML FOR GAME USER ANALYTICS.

2. DISCUSS THE BENEFITS OF INTEGRATING ONLINE ML INTO THE EXISTING APP. E.G.,
   2.1. SHOW SCENARIOS WHERE OFFLINE ML FAILS BUT ONLINE ML SUCCEEDS. (YOU CAN USE VIDEO RECORDED FOR THE PRESENTATION)
   2.2 SHOW ANOTHER IDEA FOR ANALYTICS.

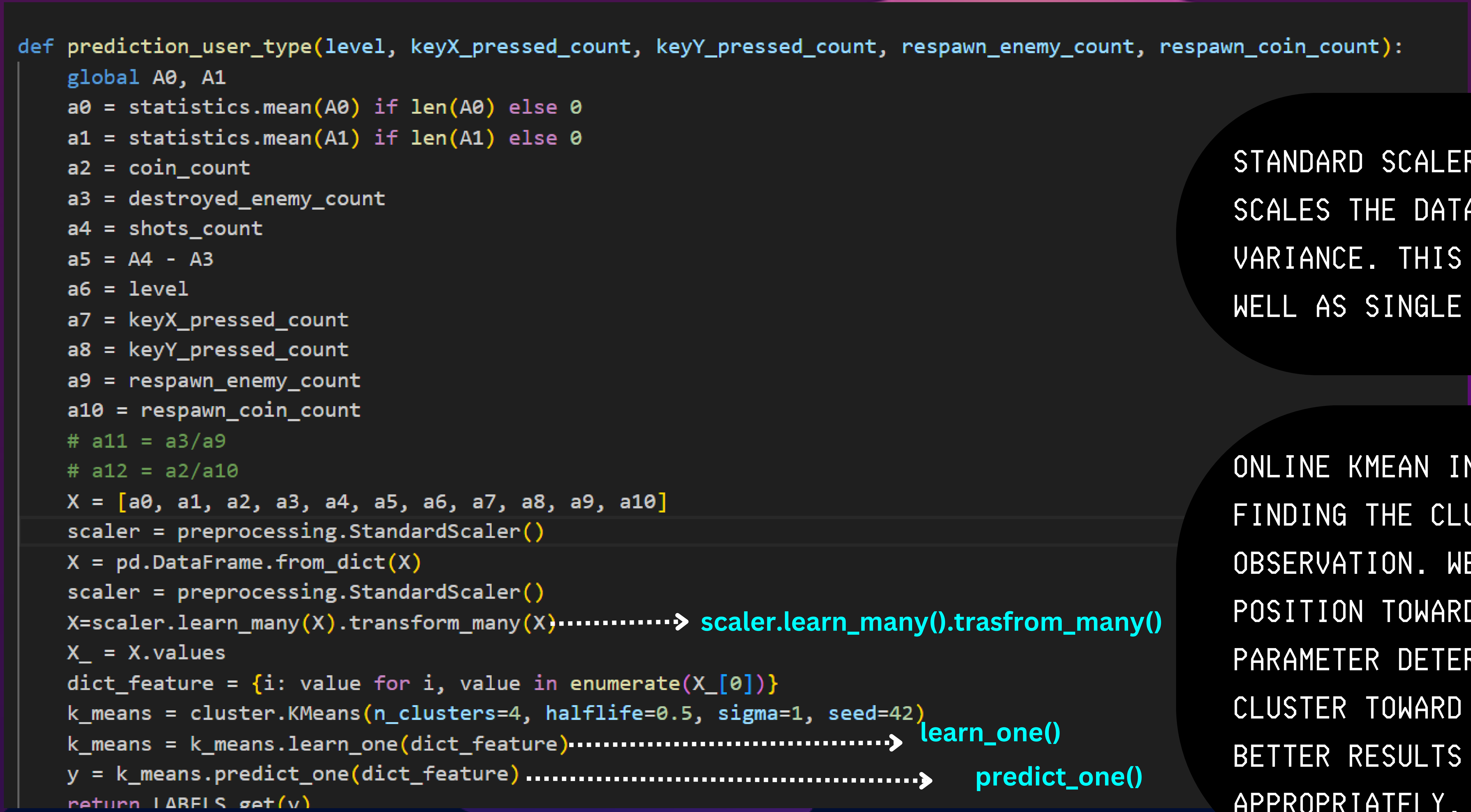
TRAIN FOR UNDERSTANDING



GAME ANALYTICS



1. INTEGRATE ONLINE ML FOR GAME USER ANALYTICS.



```
def prediction_user_type(level, keyX_pressed_count, keyY_pressed_count, respawn_enemy_count, respawn_coin_count):
    global A0, A1
    a0 = statistics.mean(A0) if len(A0) else 0
    a1 = statistics.mean(A1) if len(A1) else 0
    a2 = coin_count
    a3 = destroyed_enemy_count
    a4 = shots_count
    a5 = A4 - A3
    a6 = level
    a7 = keyX_pressed_count
    a8 = keyY_pressed_count
    a9 = respawn_enemy_count
    a10 = respawn_coin_count
    # a11 = a3/a9
    # a12 = a2/a10
    X = [a0, a1, a2, a3, a4, a5, a6, a7, a8, a9, a10]
    scaler = preprocessing.StandardScaler()
    X = pd.DataFrame.from_dict(X)
    scaler = preprocessing.StandardScaler()
    X=scaler.learn_many(X).transform_many(X)
    X_ = X.values
    dict_feature = {i: value for i, value in enumerate(X_[0])}
    k_means = cluster.KMeans(n_clusters=4, halflife=0.5, sigma=1, seed=42)
    k_means = k_means.learn_one(dict_feature)
    y = k_means.predict_one(dict_feature)
    return LABELS.get(y)
```

MODEL

STANDARD SCALER

SCALES THE DATA SO THAT IT HAS ZERO MEAN AND UNIT VARIANCE. THIS TRANSFORMER SUPPORTS MINI-BATCHES AS WELL AS SINGLE INSTANCES.

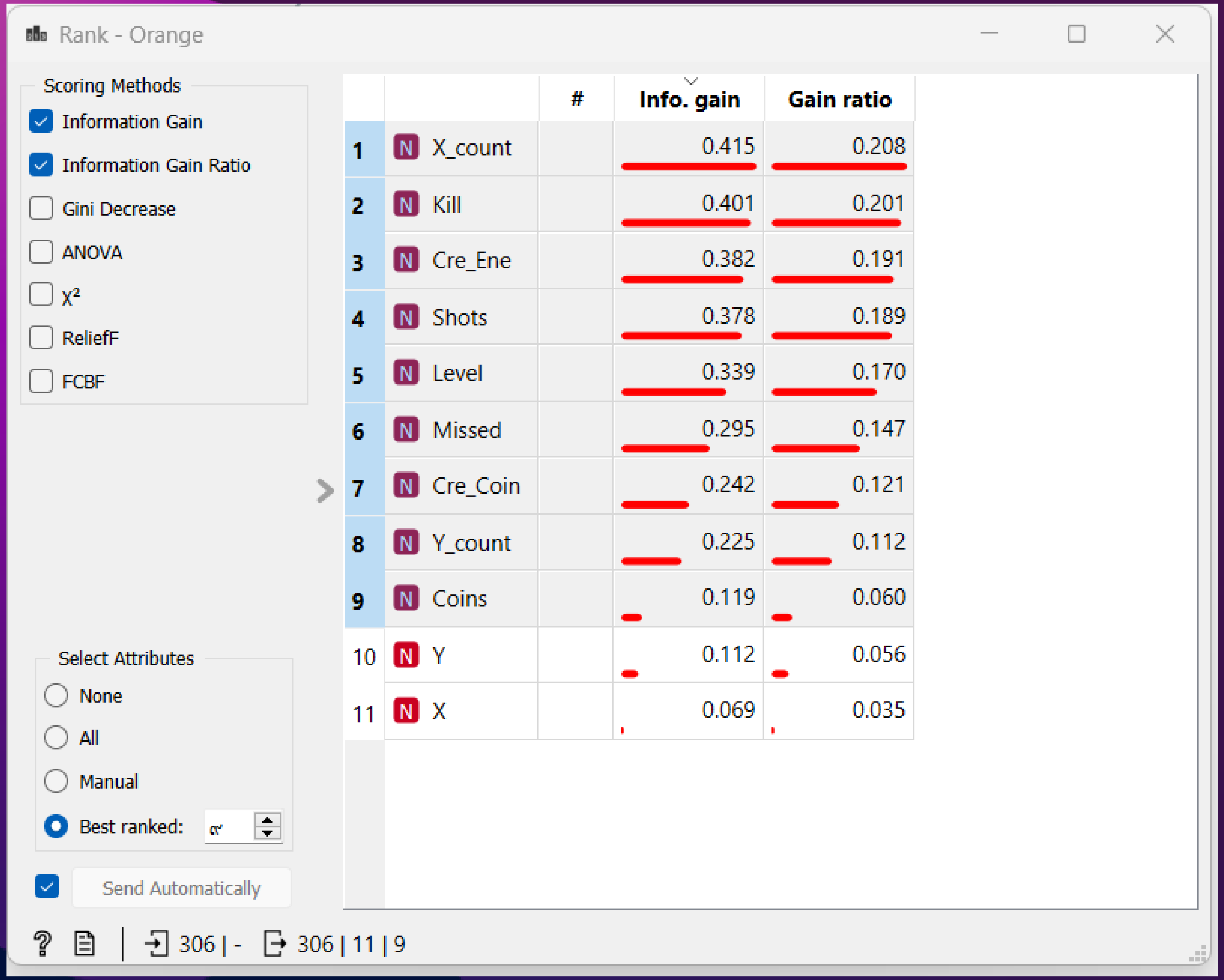
ONLINE KMEAN IN THIS IMPLEMENTATION WE START BY FINDING THE CLUSTER THAT IS CLOSEST TO THE CURRENT OBSERVATION. WE THEN MOVE THE CLUSTER'S CENTRAL POSITION TOWARDS THE NEW OBSERVATION. THE HALFLIFE PARAMETER DETERMINES BY HOW MUCH TO MOVE THE CLUSTER TOWARD THE NEW OBSERVATION. YOU WILL GET BETTER RESULTS IF YOU SCALE YOUR DATA APPROPRIATELY.

FEATURES

COLLECT THE FOLLOWING DATA EVERY 1 SECOND

A0) POSITION IN X AXIS
A1) POSITION IN Y AXIS
A2) NUMBER OF COIN COLLECTED__________TOTAL
A3) NUMBER OF DESTROY ENEMIES_________TOTAL
A4) NUMBER OF SHOTS__________________TOTAL
A5) NUMBER OF SHOTS WITHOUT ENEMIES__TOTAL
A6) LEVEL REACH______________________LATEST
A7) KEY X PRESSED COUNT______________TOTAL
A8) KEY Y PRESSED COUNT______________TOTAL
A9) NUMBER OF ENEMY CREATE___________TOTAL
A10) NUMBER OF COIN CREATE___________TOTAL

SCORING IN EACH FEATURE



Rank - Orange

Scoring Methods
- [✓] Information Gain
- [✓] Information Gain Ratio
- [ ] Gini Decrease
- [ ] ANOVA
- [ ] χ^2
- [ ] ReliefF
- [ ] FCBF

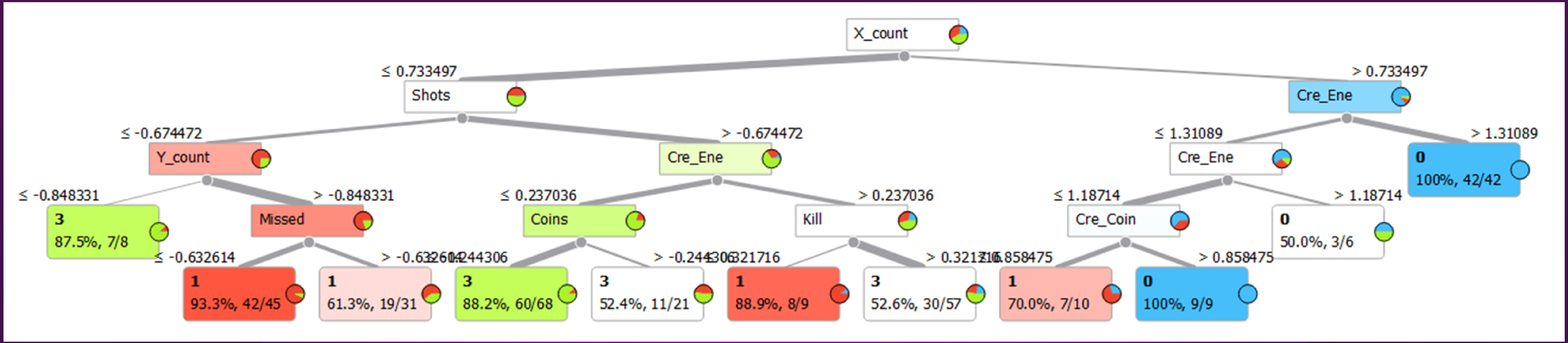
		#	Info. gain	Gain ratio
1	N X_count		0.415	0.208
2	N Kill		0.401	0.201
3	N Cre_Ene		0.382	0.191
4	N Shots		0.378	0.189
5	N Level		0.339	0.170
6	N Missed		0.295	0.147
7	N Cre_Coin		0.242	0.121
8	N Y_count		0.225	0.112
9	N Coins		0.119	0.060
10	N Y		0.112	0.056
11	N X		0.069	0.035

Select Attributes
- ○ None
- ○ All
- ○ Manual
- ● Best ranked: 9

[✓] Send Automatically

306 | -   306 | 11 | 9

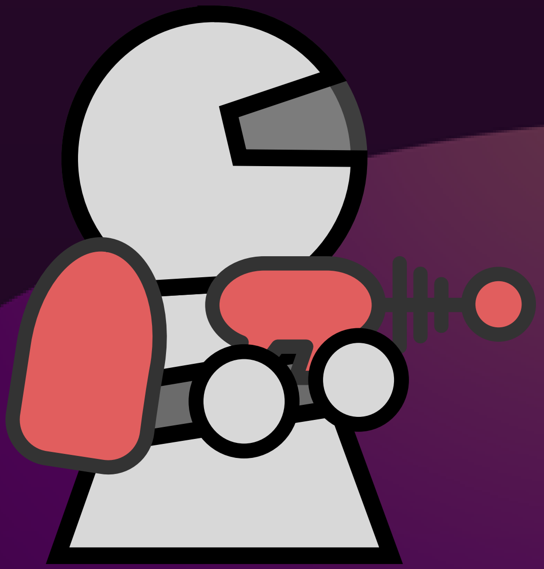
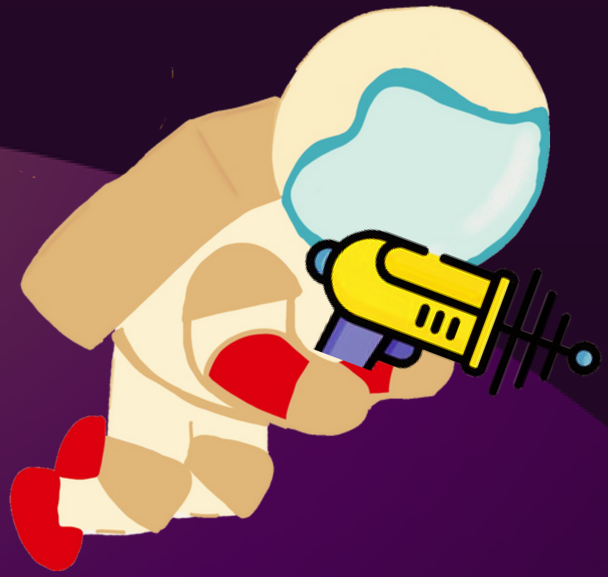
DICISION TREE



CLUSTER 0: “SURVIVALIST“: HIGHLIGHTS THE PLAYER'S FOCUS ON COLLECTING POWER-UPS (COINS) WHILE BEING MORE PRONE TO RISKING ITS OWN LIFE.
BEHAVIOR: THIS PLAYER TYPE EXHIBITS HIGH MOVEMENT ALONG THE X-AXIS, COMBINED WITH FREQUENT SHOOTING. IT PRIORITIZES COLLECTING COINS WHILE MAINTAINING A LONGER LIFETIME.

CLUSTER 1: “AGGRESSIVE SHOOTER“: REFLECTS THE PLAYER'S AGILITY AND SHOOTING ABILITY.
BEHAVIOR: THIS PLAYER TYPE FOCUSES ON HIGH MOVEMENT ALONG THE Y-AXIS, WITH PRECISE SHOOTING AT A LOWER RATE. IT TENDS TO HAVE HIGH MANEUVERABILITY AND AVOIDS GETTING HIT.

CLUSTER 3:: “ACHIEVER“: " HIGHLIGHTS THE PLAYER'S FOCUS ON COLLECTING POWER-UPS (COINS) WHILE BEING MORE PRONE TO RISKING ITS OWN LIFE.
BEHAVIOR: THIS PLAYER TYPE EXHIBITS HIGH MOVEMENT ALONG THE Y-AXIS, COMBINED , WITH PRECISE SHOOTING AT A HIERER RATE .IT PRIORITIZES COLLECTING COINS WHILE MAINTAINING A LOWER LIFETIME.

0 SURVIVALIST

Behavior: This player type exhibits high movement along the x-axis, combinexd with frequent shooting. It prioritizes collecting coins while maintaining a longer lifetime.

1 AGGRESSIVE SHOOTER

Behavior: This player type focuses on high movement along the Y-axis, with precise shooting at a lower rate. It tends to have high maneuverability and avoids getting hit.

2 CASUAL

Behavior: This normal player type

3 ARCHIVER

Behavior: This player type exhibits high movement along the Y-axis, combined , with precise shooting at a HIERer rate .It prioritizes collecting coins while maintaining a lower lifetime.

DISCUSS THE BENEFITS OF INTEGRATING ONLINE ML INTO THE EXISTING APP

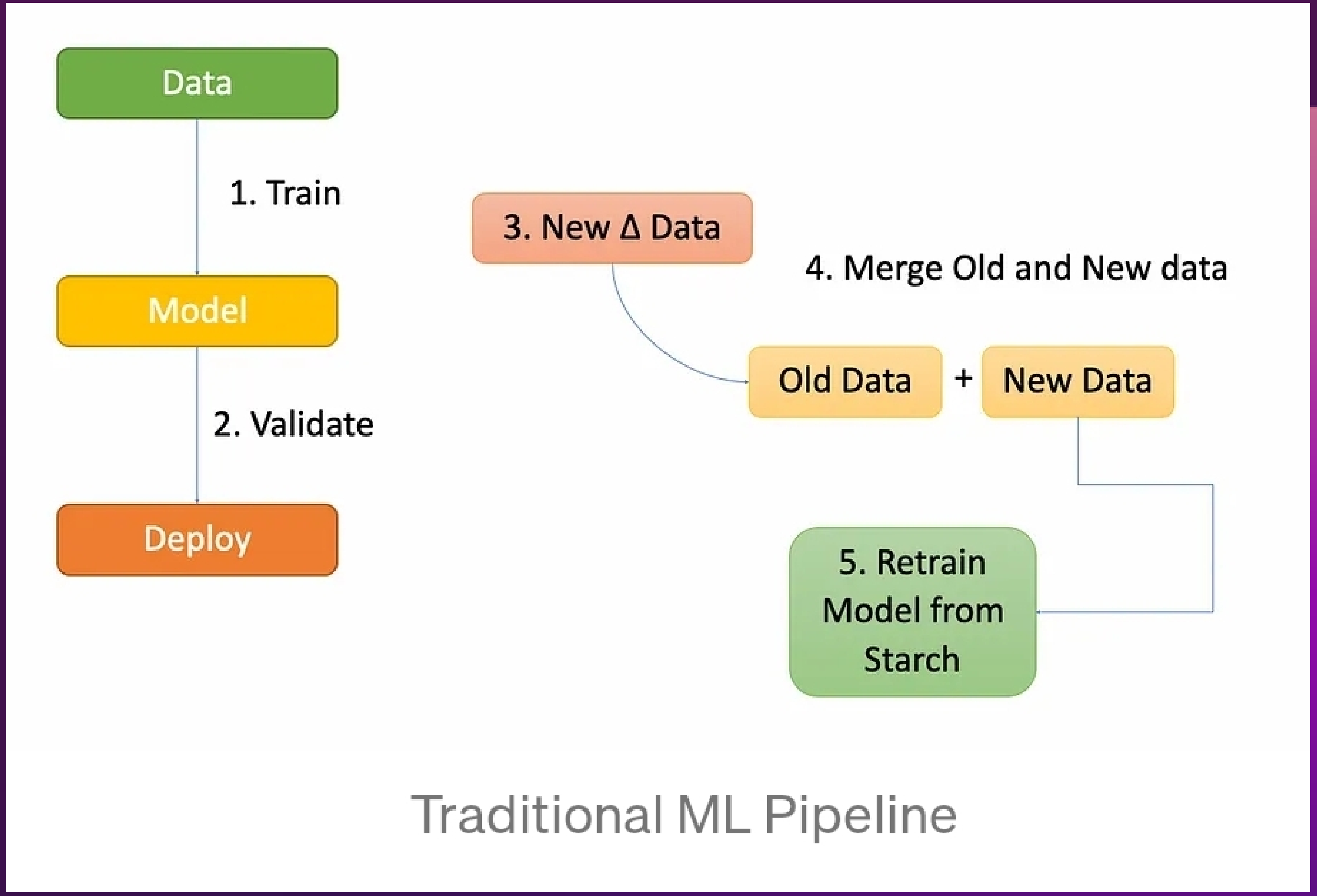
BENEFIT ONLINE ML VS OFFLINE MACHINE LEARNING

STANDARDSCALER

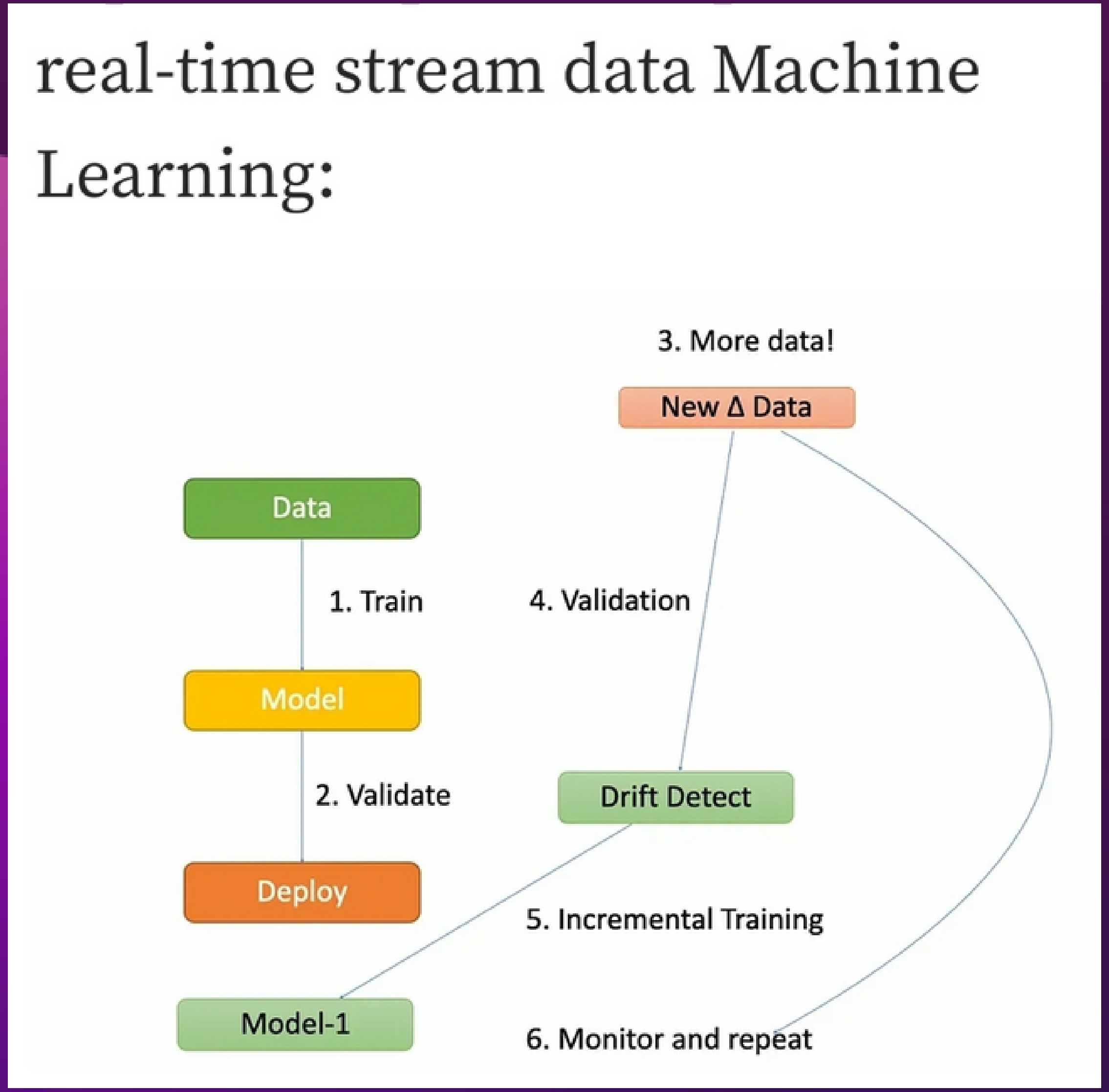
ADAPTIVE GAMEPLAY

- INCREMENTAL K-MEANS

OFF LINE ML VS ONLINE ML



Traditional ML Pipeline



real-time stream data Machine Learning:

IF INSTEAD OF THE ORIGINAL DATASET BEING OF 100 DATA POINTS IMAGINE IT IS 10 MILLION DATA POINTS AND THE NEWLY GENERATED DELTA HAS 2 MILLION DATA POINTS. NOW YOU CAN SEE HOW IT IS HARD TO SCALE, ALSO IT IS NOT A TIME AND COMPUTATIONALLY EFFICIENT METHOD?

THE MACHINE LEARNING ALGORITHMS FOR REAL-TIME STREAM DATA LEARN FROM THE INCOMING DELTA DATA. HERE THE MODEL CONTINUES TO LEARN FROM NEW DATA POINTS AND OPTIMIZES ITS OBJECTIVE FUNCTION.

https://kvirajdatt.medium.com/incremental-machine-learning-for-streaming-data-with-river

OFF LINE ML VS ONLINE ML

SOME CHALLENGES FOR INCREMENTAL LEARNING ON STREAMING DATA:

- *USUALLY, WITH STREAMING DATA, THE DATA IS NOT SAVED FOR A LONGER TIME. HENCE THE LEARNING ALGORITHMS CANNOT GET MULTIPLE PASSES THROUGH THE DATA. DATA MUST BE PROCESSED IN A SINGLE STEP.*

- *HIGHER MEMORY AND PROCESSING REQUIREMENTS.*

- *FOR A CLASSIFICATION TASK, ALL THE CLASSES FOR THE TARGET VARIABLE MAY NOT BE AVAILABLE RIGHT UP FRONT AS USUALLY IS THE CASE IN TRADITIONAL BATCH PROCESSING MACHINE LEARNING. EX: CONSIDER YOUR TARGET VARIABLE IS PREDICTING THE COLOR OF THE CAR THE CUSTOMER SELECTS. WITH STREAM DATA, YOU EXACTLY DON'T KNOW HOW MANY CLASSES OF COLOR YOU WILL END UP WITH.*

https://kvirajdatt.medium.com/incremental-machine-learning-for-streaming-data-with-river

BENEFIT ONLINE ML VS OFFLINE MACHINE LEARNING

STANDARDSCALER RIVER MACHINE LEARNING

SCALES THE DATA SO THAT IT HAS ZERO MEAN AND UNIT VARIANCE. UNDER THE HOOD, A RUNNING MEAN AND A RUNNING VARIANCE ARE MAINTAINED. THE SCALING IS SLIGHTLY DIFFERENT THAN WHEN SCALING THE DATA IN BATCH BECAUSE THE EXACT MEANS AND VARIANCES ARE NOT KNOWN IN ADVANCE. HOWEVER, THIS DOESN'T HAVE A DETRIMENTAL IMPACT ON PERFORMANCE IN THE LONG RUN.

```
SKLEARN DATAFRAME    2.347152  0.912167  0.882779 -0.027435 -0.146442 -0.452538  1.367096
```

PREDICT CLASS = 3

```
SKLEARN 1 ROW        2.161454  2.045317 -0.571490 -0.485296 -0.468057 -0.657684  -0.640445
```

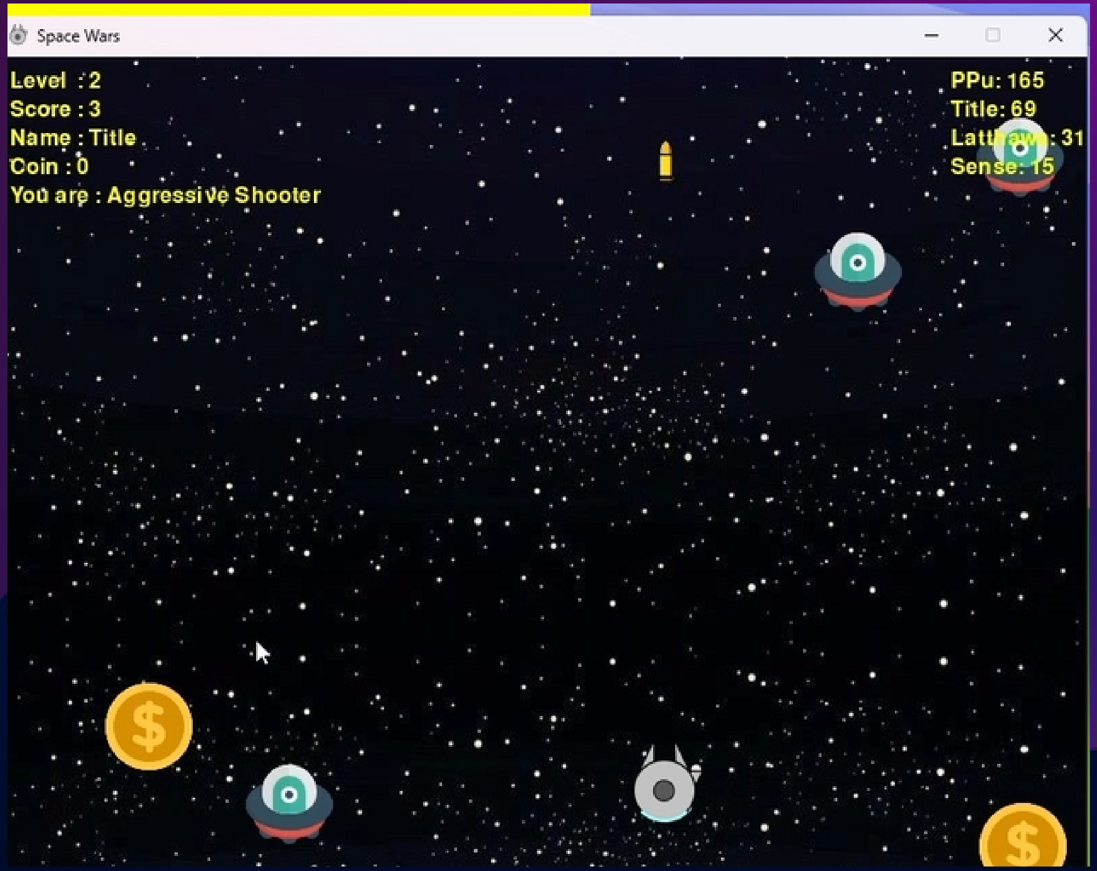
PREDICT CLASS = 0

BENEFIT ONLINE VS OFFLINE MACHINE LEARNING

ADAPTIVE GAMEPLAY:

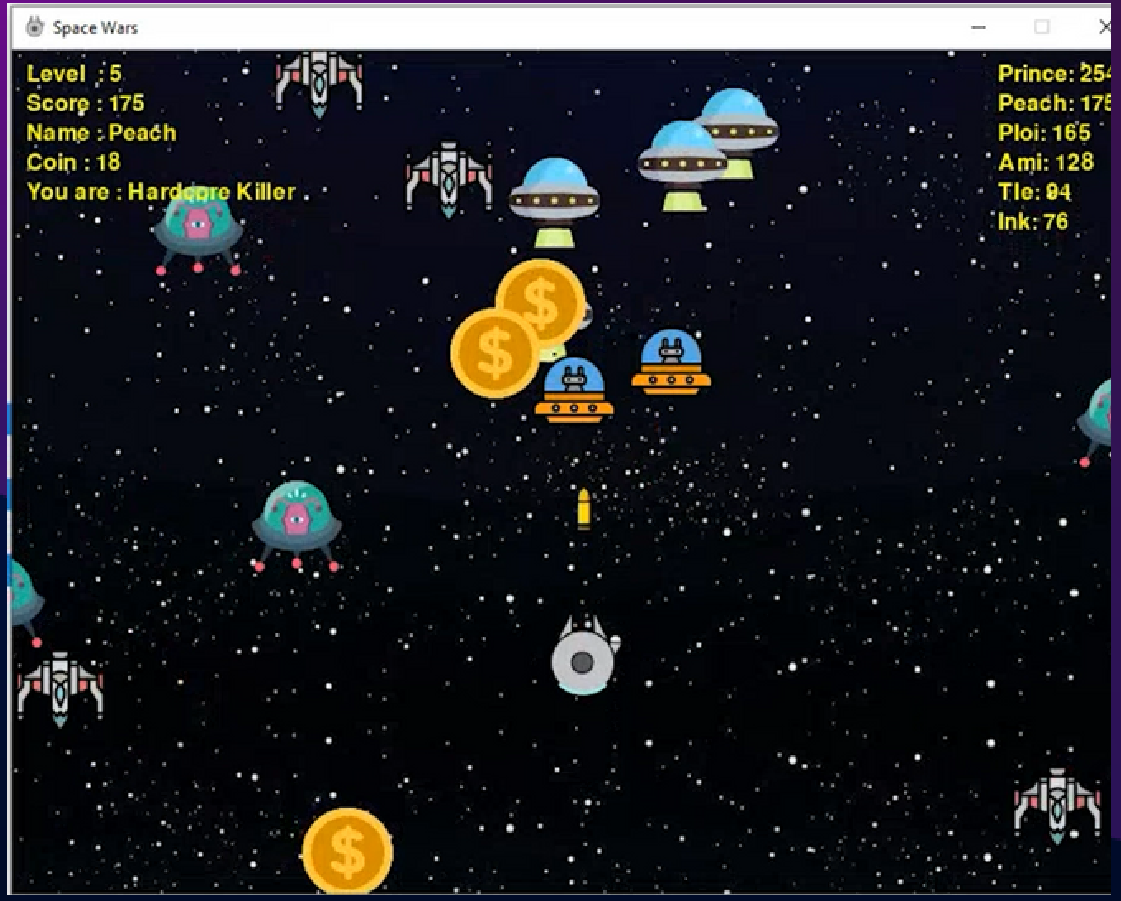
OFFLINE MACHINE LEARNING MODELS RELY ON HISTORICAL DATA AND MAY STRUGGLE TO CAPTURE THE DYNAMIC NATURE OF GAMEPLAY. CONVERSELY, ONLINE ML ALGORITHMS CAN CONTINUOUSLY LEARN FROM REAL-TIME PLAYER INTERACTIONS, ENABLING THEM TO ADAPT TO INDIVIDUAL PREFERENCES AND DYNAMICALLY ADJUST THE GAMEPLAY EXPERIENCE. FOR INSTANCE, IF A PLAYER FREQUENTLY SHOOTS AT THE BEGINNING OF THE GAME, THE ONLINE ML ALGORITHM CAN INFER THAT THEY PREFER AN AGGRESSIVE PLAYSTYLE, LEADING TO A MORE FAST-PACED AND INTENSE GAMING EXPERIENCE FOR THAT PLAYER.

ONLINE : LEVEL 2



```
Level  :2
Score : 3
Name : Title
Coin : 0
You are : Aggressive Shooter

PPu: 165
Title: 69
Latt[illegible]: 31
Sense: 15
```

OFFLINE : LEVEL 5



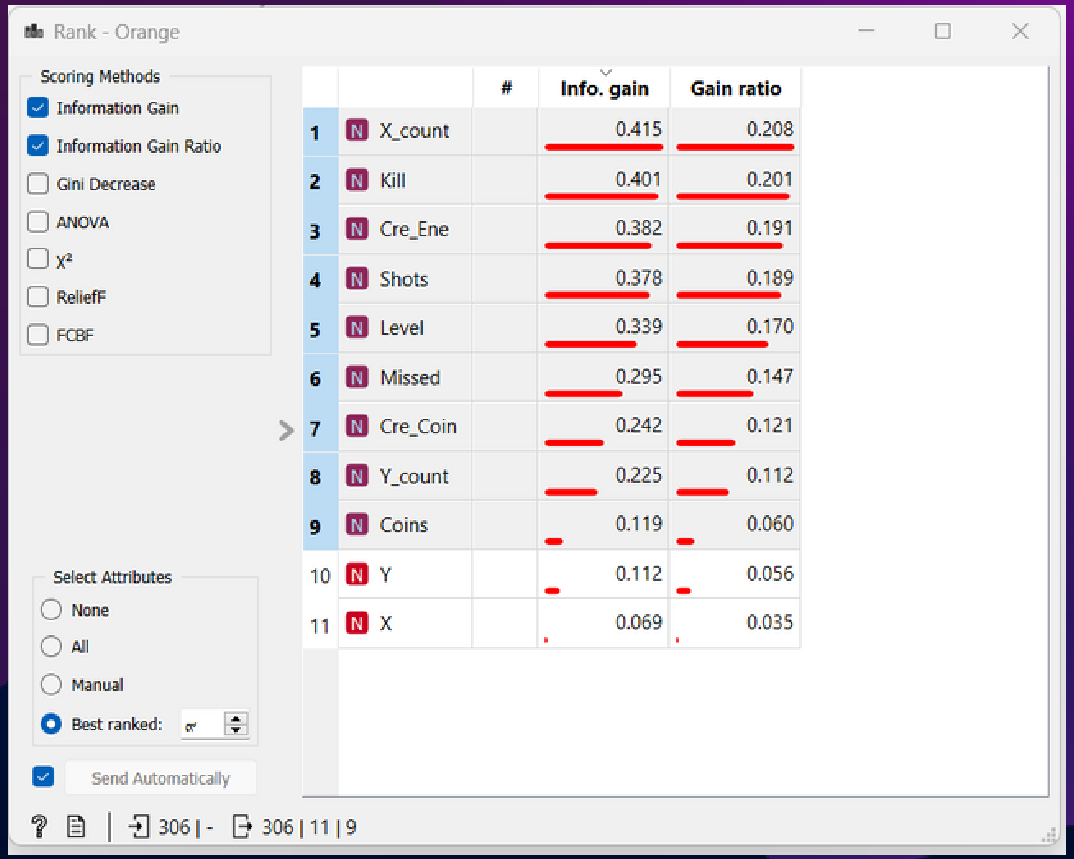
```
Level  : 5
Score : 175
Name : Peach
Coin : 18
You are : Hardcore Killer

Prince: 25
Peach: 17
Ploi: 165
Ami: 128
Tle: 94
Ink: 76
```

SHOW ANOTHER IDEA FOR ANALYTICS.

OUR ANALYTICS APPROACH INVOLVES USING MULTIPLE TECHNIQUES TO GAIN COMPREHENSIVE INSIGHTS FROM DATA.

WE EMPLOY K-MEANS CLUSTERING TO CATEGORIZE DATA POINTS INTO CLUSTERS, ASSIGNING LABELS TO EACH CLUSTER.

TO FURTHER ANALYZE THE CLUSTERS, WE UTILIZE DECISION TREES AND RANDOM FORESTS. RANDOM FOREST HELP US IDENTIFY FEATURE IMPORTANCE RANKINGS. AND DECISION TREES FOR INFLUENCE THE BEHAVIOR OF EACH CLUSTER, PROVIDING A CLEAR AND INTERPRETABLE REPRESENTATION.



Rank - Orange

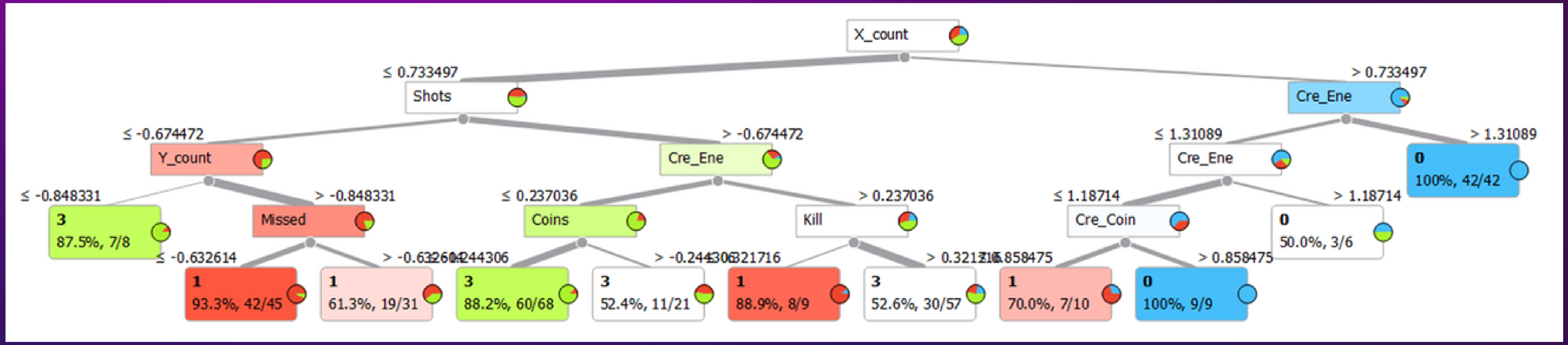
Scoring Methods
- [✓] Information Gain
- [✓] Information Gain Ratio
- [ ] Gini Decrease
- [ ] ANOVA
- [ ] χ²
- [ ] ReliefF
- [ ] FCBF

		#	Info. gain	Gain ratio
1	N X_count		0.415	0.208
2	N Kill		0.401	0.201
3	N Cre_Ene		0.382	0.191
4	N Shots		0.378	0.189
5	N Level		0.339	0.170
6	N Missed		0.295	0.147
7	N Cre_Coin		0.242	0.121
8	N Y_count		0.225	0.112
9	N Coins		0.119	0.060
10	N Y		0.112	0.056
11	N X		0.069	0.035

Select Attributes
- ○ None
- ○ All
- ○ Manual
- ● Best ranked:

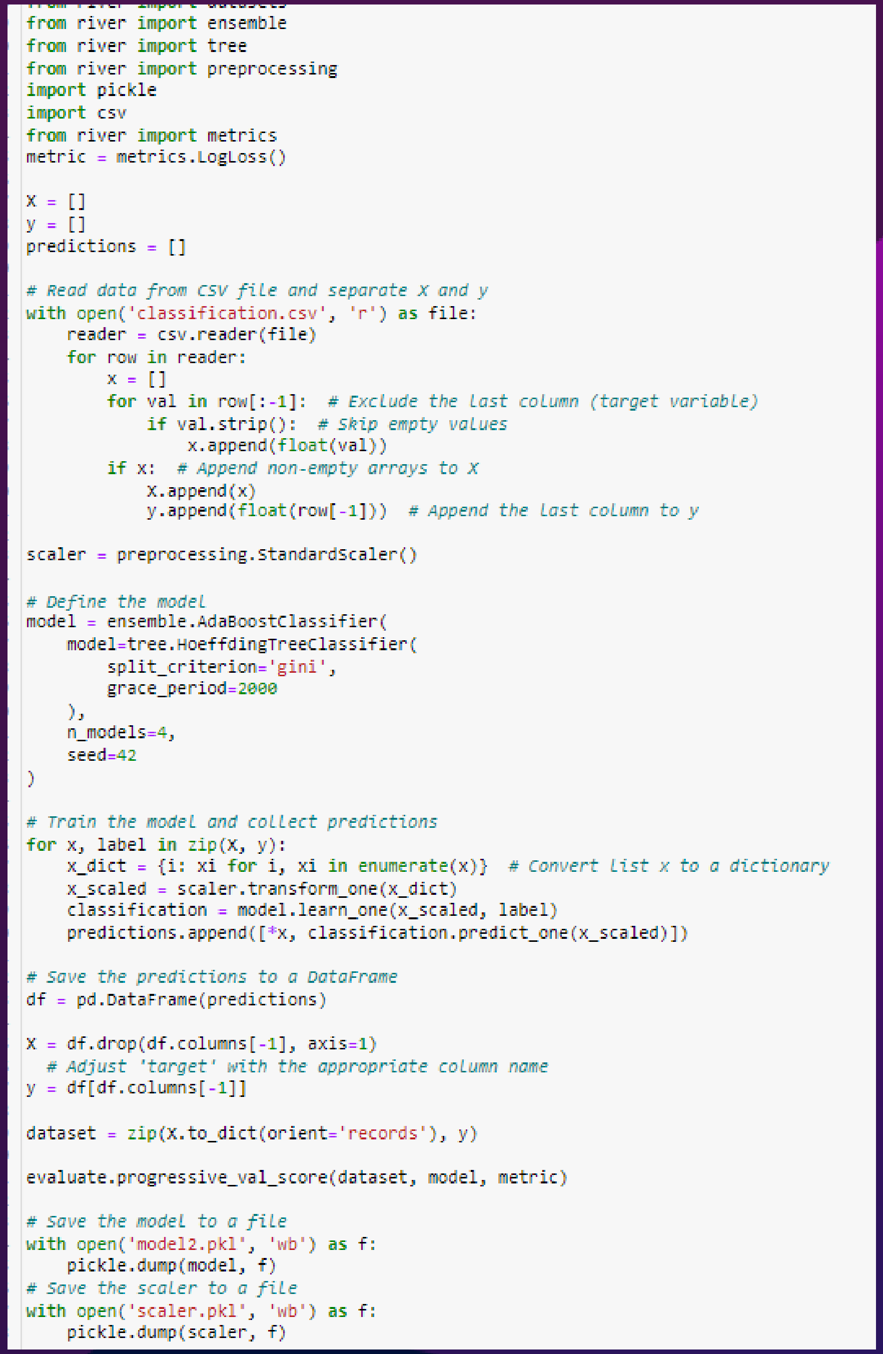
[✓] Send Automatically

306 | - 306 | 11 | 9

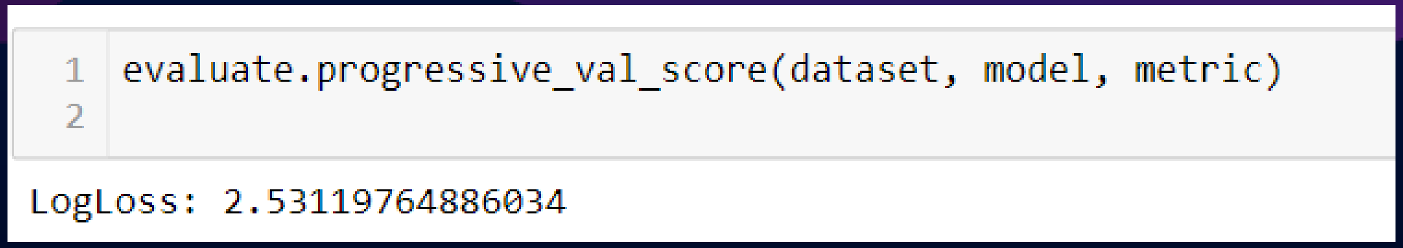


SHOW ANOTHER IDEA FOR ANALYTICS.

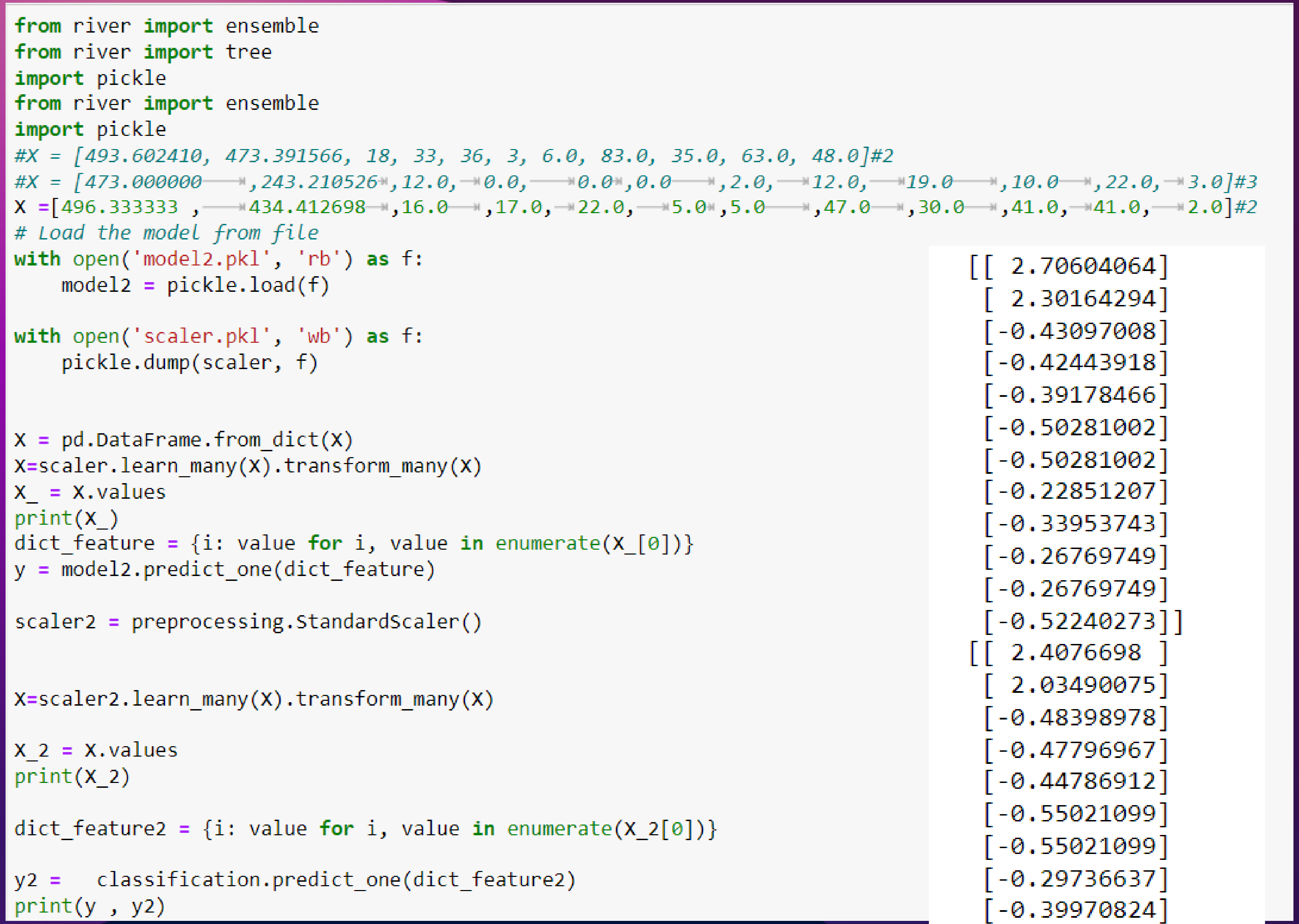
ONLINE CLASSIFICATION MACHINE LEARNING



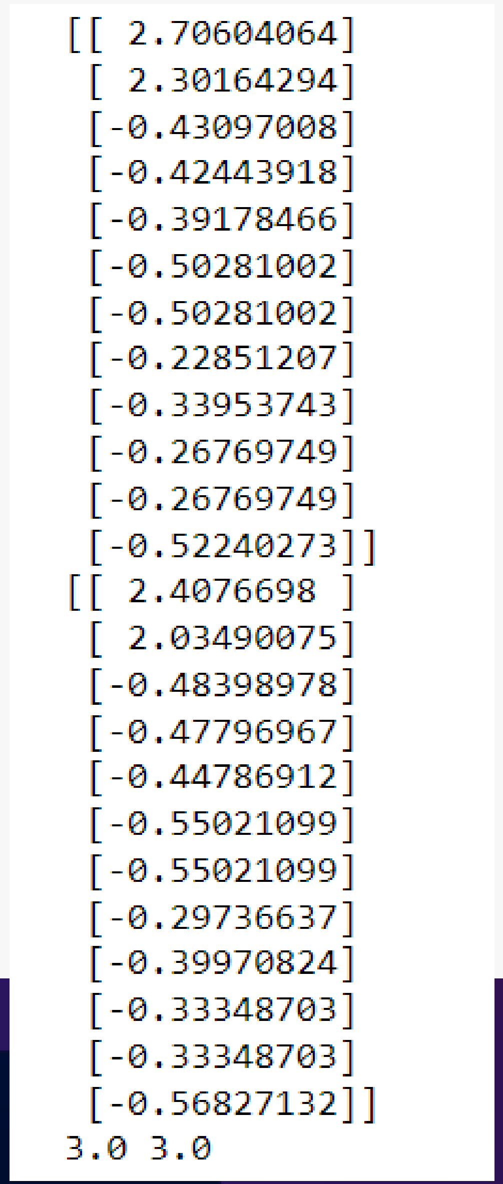
```
from river import ensemble
from river import tree
from river import preprocessing
import pickle
import csv
from river import metrics
metric = metrics.LogLoss()

X = []
y = []
predictions = []

# Read data from CSV file and separate X and y
with open('classification.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        x = []
        for val in row[:-1]:  # Exclude the last column (target variable)
            if val.strip():  # Skip empty values
                x.append(float(val))
        if x:  # Append non-empty arrays to X
            X.append(x)
            y.append(float(row[-1]))  # Append the last column to y

scaler = preprocessing.StandardScaler()

# Define the model
model = ensemble.AdaBoostClassifier(
    model=tree.HoeffdingTreeClassifier(
        split_criterion='gini',
        grace_period=2000
    ),
    n_models=4,
    seed=42
)

# Train the model and collect predictions
for x, label in zip(X, y):
    x_dict = {i: xi for i, xi in enumerate(x)}  # Convert list x to a dictionary
    x_scaled = scaler.transform_one(x_dict)
    classification = model.learn_one(x_scaled, label)
    predictions.append([*x, classification.predict_one(x_scaled)])

# Save the predictions to a DataFrame
df = pd.DataFrame(predictions)

X = df.drop(df.columns[-1], axis=1)
  # Adjust 'target' with the appropriate column name
y = df[df.columns[-1]]

dataset = zip(X.to_dict(orient='records'), y)

evaluate.progressive_val_score(dataset, model, metric)

# Save the model to a file
with open('model2.pkl', 'wb') as f:
    pickle.dump(model, f)
# Save the scaler to a file
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)
```



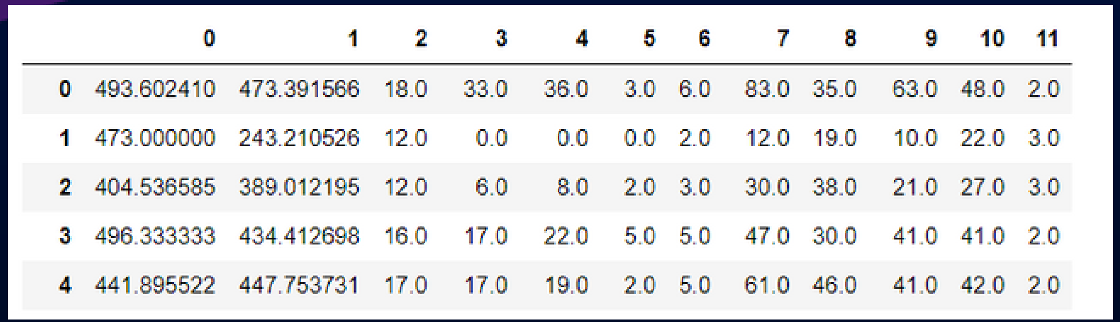
```
evaluate.progressive_val_score(dataset, model, metric)

LogLoss: 2.53119764886034
```



```
from river import ensemble
from river import tree
import pickle
from river import ensemble
import pickle
#X = [493.602410, 473.391566, 18, 33, 36, 3, 6.0, 83.0, 35.0, 63.0, 48.0]#2
#X = [473.000000   ,243.210526 ,12.0,  0.0,     0.0 ,0.0    ,2.0,   12.0,   19.0    ,10.0   ,22.0,  3.0]#3
X =[496.333333 ,    434.412698  ,16.0   ,17.0,  22.0,   5.0 ,5.0     ,47.0   ,30.0   ,41.0,  41.0,   2.0]#2
# Load the model from file
with open('model2.pkl', 'rb') as f:
    model2 = pickle.load(f)

with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)


X = pd.DataFrame.from_dict(X)
X=scaler.learn_many(X).transform_many(X)
X_ = X.values
print(X_)
dict_feature = {i: value for i, value in enumerate(X_[0])}
y = model2.predict_one(dict_feature)

scaler2 = preprocessing.StandardScaler()


X=scaler2.learn_many(X).transform_many(X)

X_2 = X.values
print(X_2)

dict_feature2 = {i: value for i, value in enumerate(X_2[0])}

y2 =   classification.predict_one(dict_feature2)
print(y , y2)
```



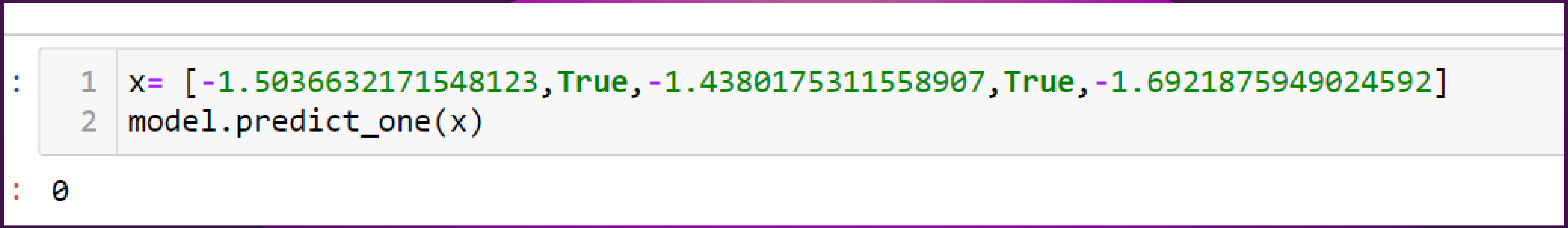
```
[[ 2.70604064]
 [ 2.30164294]
 [-0.43097008]
 [-0.42443918]
 [-0.39178466]
 [-0.50281002]
 [-0.50281002]
 [-0.22851207]
 [-0.33953743]
 [-0.26769749]
 [-0.26769749]
 [-0.52240273]]
[[ 2.4076698 ]
 [ 2.03490075]
 [-0.48398978]
 [-0.47796967]
 [-0.44786912]
 [-0.55021099]
 [-0.55021099]
 [-0.29736637]
 [-0.39970824]
 [-0.33348703]
 [-0.33348703]
 [-0.56827132]]
3.0 3.0
```



	0	1	2	3	4	5	6	7	8	9	10	11
0	493.602410	473.391566	18.0	33.0	36.0	3.0	6.0	83.0	35.0	63.0	48.0	2.0
1	473.000000	243.210526	12.0	0.0	0.0	0.0	2.0	12.0	19.0	10.0	22.0	3.0
2	404.536585	389.012195	12.0	6.0	8.0	2.0	3.0	30.0	38.0	21.0	27.0	3.0
3	496.333333	434.412698	16.0	17.0	22.0	5.0	5.0	47.0	30.0	41.0	41.0	2.0
4	441.895522	447.753731	17.0	17.0	19.0	2.0	5.0	61.0	46.0	41.0	42.0	2.0

SHOW ANOTHER IDEA FOR ANALYTICS.

ONLINE CLASSIFICATION MACHINE LEARNING



```
x= [-1.5036632171548123,True,-1.4380175311558907,True,-1.6921875949024592]
model.predict_one(x)
```

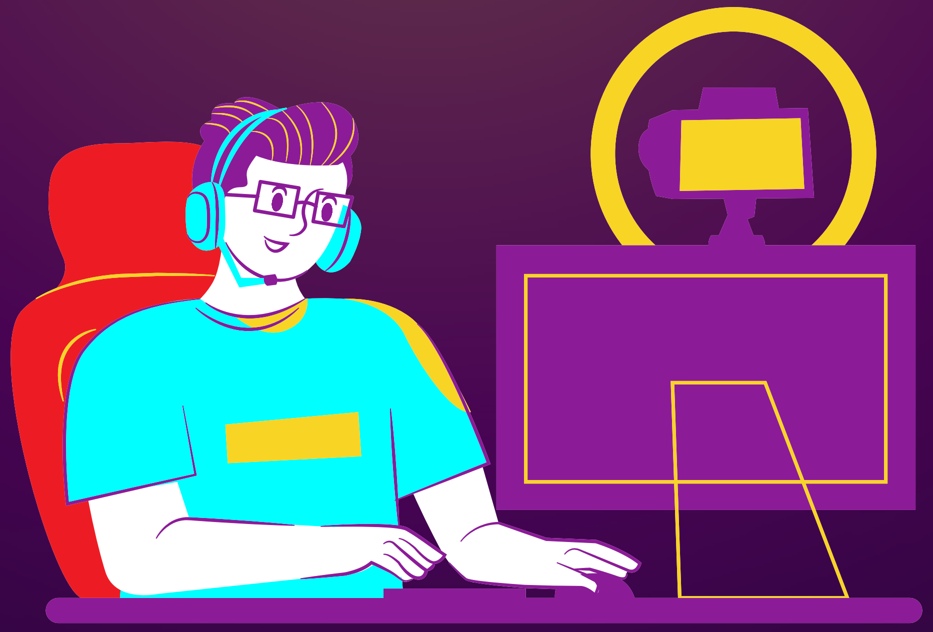
```
0
```

SHOW ANOTHER IDEA FOR ANALYTICS.

OUR FUTURE PLAN INVOLVES IMPLEMENTING A REAL-TIME FRAUD DETECTION SYSTEM FOR A SPACE WARSHIP GAME ANALYTICS USING ONLINE MACHINE LEARNING WITH THE RIVER FRAMEWORK. BY LEVERAGING THE POWER OF ONLINE MACHINE LEARNING ALGORITHMS, WE AIM TO CONTINUOUSLY MONITOR AND DETECT FRAUDULENT ACTIVITIES IN THE GAME IN REAL TIME.THE SYSTEM WILL ANALYZE VARIOUS GAME METRICS, PLAYER INTERACTIONS, AND IN-GAME TRANSACTIONS TO IDENTIFY SUSPICIOUS PATTERNS INDICATIVE OF FRAUDULENT BEHAVIOR. IT WILL ADAPT AND LEARN FROM REAL-TIME DATA, ALLOWING FOR DYNAMIC ADJUSTMENTS AND IMPROVED ACCURACY IN FRAUD DETECTION.

THE ONLINE MACHINE LEARNING ALGORITHMS WILL CONTINUOUSLY UPDATE AND EVOLVE AS NEW DATA ARRIVES, ENABLING THE SYSTEM TO STAY UP-TO-DATE WITH EMERGING FRAUD TECHNIQUES. WE CAN ENHANCE THE OVERALL GAME EXPERIENCE BY ENSURING FAIR PLAY, PROTECTING PLAYERS FROM FRAUDULENT ACTIVITIES, AND MAINTAINING THE INTEGRITY OF THE GAME'S ECONOMY.

THIS PROACTIVE APPROACH WILL CONTRIBUTE TO A MORE SECURE AND ENJOYABLE GAMING ENVIRONMENT FOR ALL PLAYERS INVOLVED.



DEMO

MEMBER

6420422002	ธนากร วิธุรัติ
6420422008	วรรณนภา ศรีเพ็ญ
6420422011	จันทรท์นีย์ พัฒนสุขกุล
6420422017	วิศรุต วงศ์ซิ้ม
6420422021	สุชาวลี จีระธัญญาสกุล