



Alaba Adewunmi
Alireza Eftekhari
Aigiun Guseinova
Chakrapani Suresh

California State University,

Long Beach

5/3/2023

TABLE OF CONTENTS

Company Overview

Business Model Canvas

Value Proposition Canvas

Hypotheses

Methods

Monetization

Product Prototype

Company Overview

Founded in 2022 by four MSIS students from California State University, Long Beach, **MetaZee** provides a digital collectibles betting platform based on the Zee blockchain. With nonfungible tokens, the platform offers a collective fantasy soccer experience allowing players to manage their players and earn prizes as well as perform betting to get earnings. Furthermore, we would be integrating Metaverse technology to create a more fascinating experience for our fans to experience the game. We achieve this by naming our company MetaZee: thus, joining **Meta** experience with **Zee** (our own currency).



symbol for Zee currency (coin)

MetaZee aims to revolutionize the gaming and financial services industry through innovative betting model. The company's platform helps in quick transactions without using costly intermediaries. Moreover, we would be offering tools for real-time transaction data, crypto information, and analysis, providing investors and developers with an easy and safe way to buy, sell and store cryptocurrency. **MetaZee** has a competitive advantage over its rivals and offers anonymity. A customer does not need to provide a country, postal code, and phone number just to open an account and deposit crypto, as well as when you withdraw your funds. Betting will be implemented throughout our own system (website and application) and would aim to become the best crypto football betting sites in 2023.

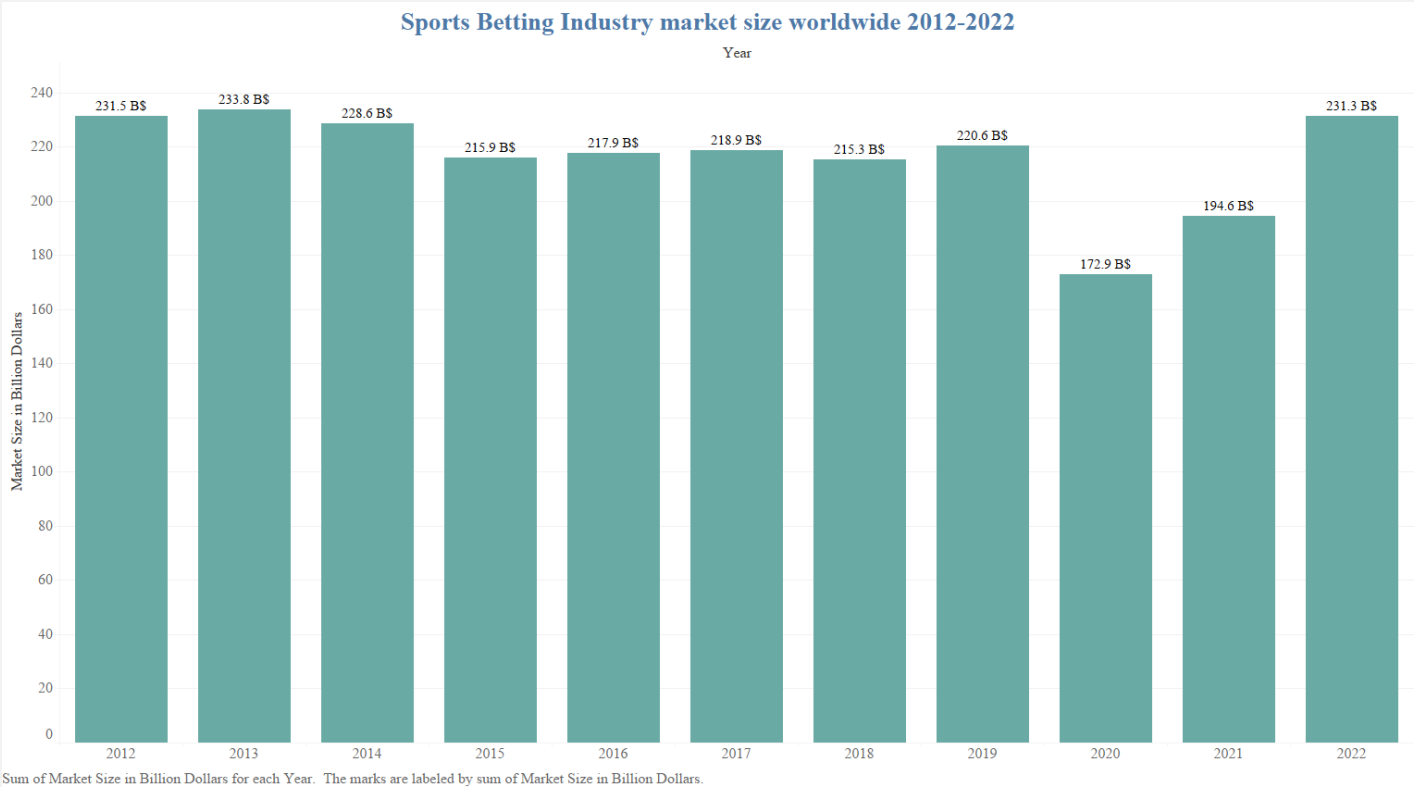
The company has 40 employees all over the world with various locations in Europe, Asia, North and South America.

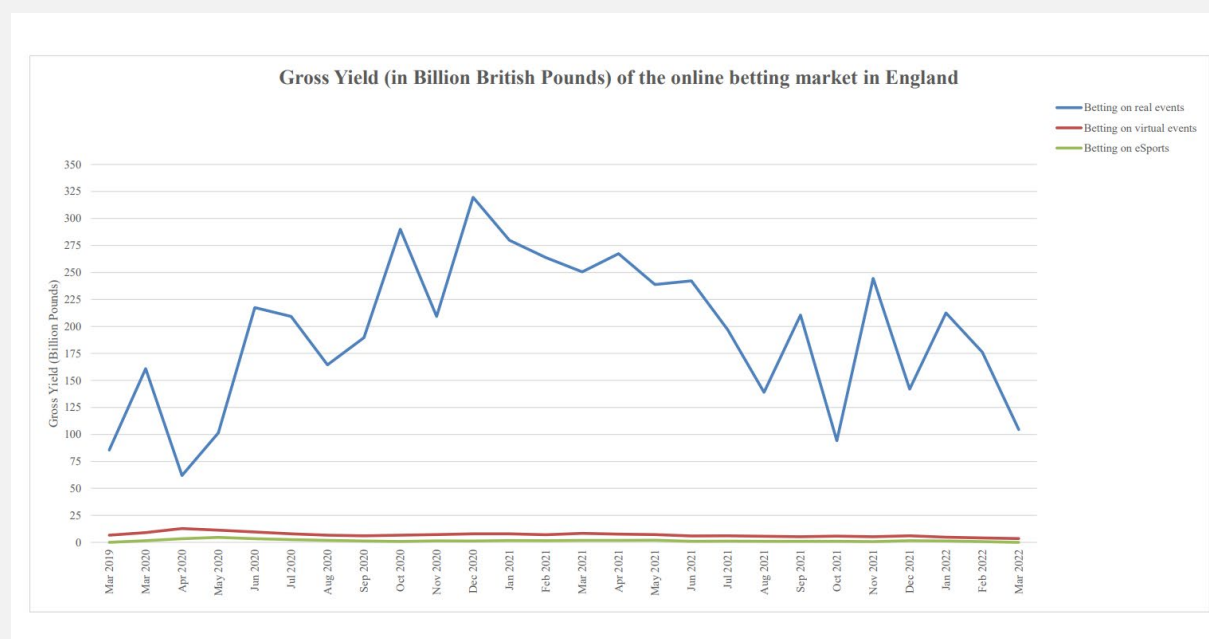
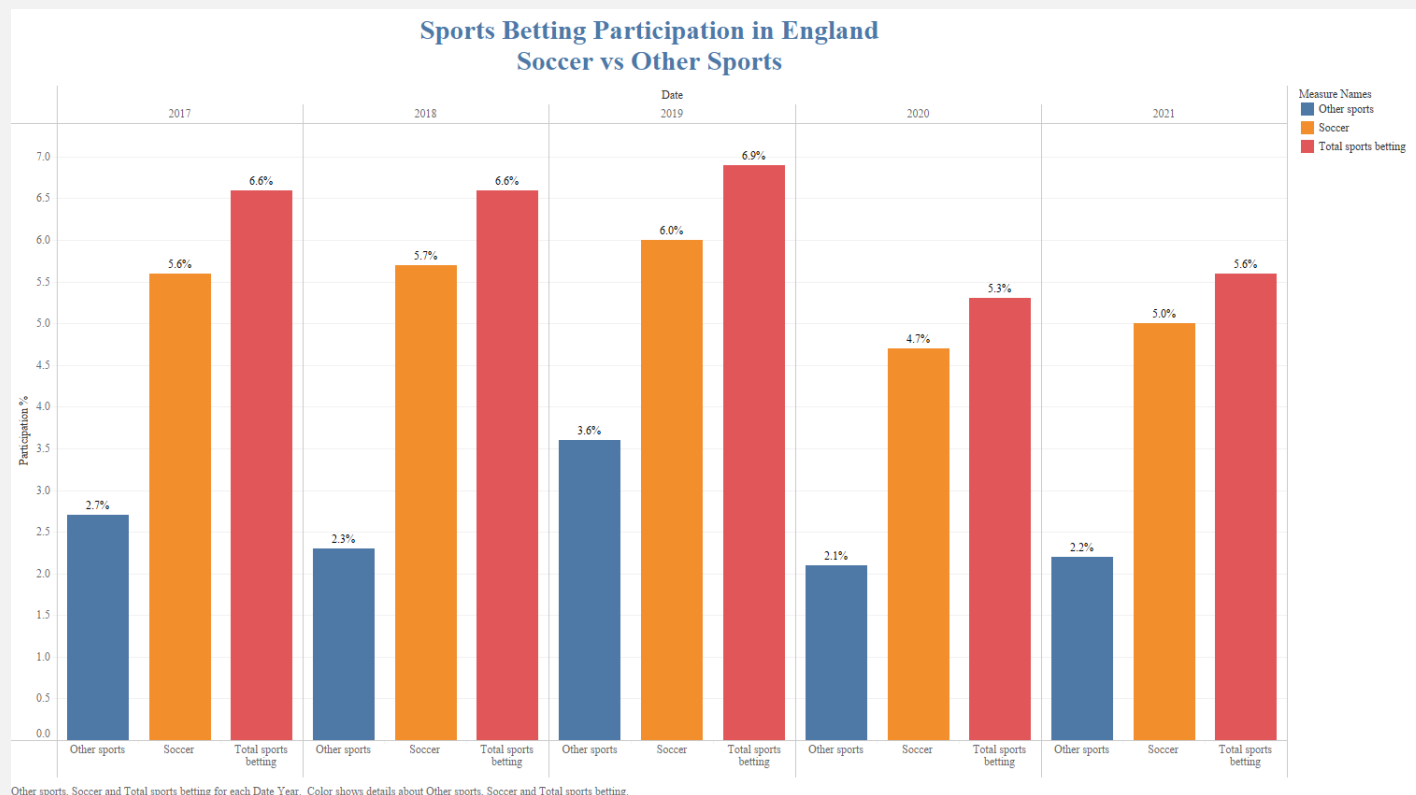
Founders: Aigiun Guseinova, Alaba Adewunmi, Alireza Eftekhari, Chakrapani Suresh

Last Funding Type: Venture

MetaZee HQ is located in London, England, United Kingdom.

MetaZee, a start-up blockchain-based soccer betting platform, has raised \$5 million from high-profile investors backing major companies like Meta, Amazon, and Netflix.





MetaZee- a spiral - shaped ball logo has been demonstrated below.

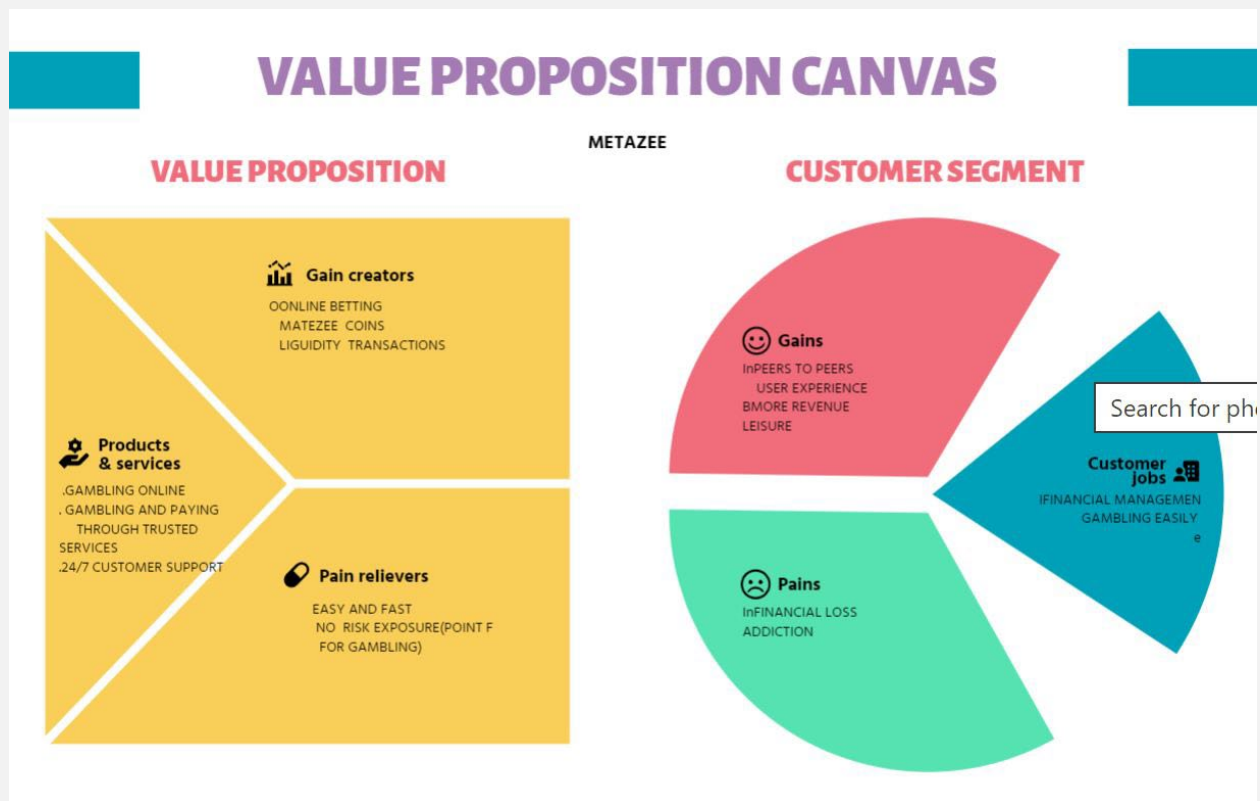
Slogan: **BET THE WORLD** indicates the global scale of operations.



Value Propositions

The **main problems** that our company is willing to solve:

- Becoming a leading global betting company
- Exciting experience from the use of our services
- Global availability to enjoy our services from anywhere in the world
- Peer-to-peer betting exchange
- Offer of liquidity on transactions
- No risk associated with bets (risk-free environment)



MetaZee has been holding its **key activities** to produce, market and deliver the solutions in the following spheres such as:

- Marketing and Advertising
- User Experience (UX)
- Software Development
- Customer Support
- Global Networking
- Global Community

Our company will be utilizing **global resources** to deliver best solutions to the global market:

- Reputation and image of the industry
- In house technology

- MetaZee was built on a stock exchange model
- Innovative products
- Creative Marketing

The key partners to be involved in the business are mentioned below:

- Sport event organizers
- Partners (Vendors and suppliers)
- Betting and Booking companies
- Strategic and Alliance partners
- Gambling locations

Monetization

Blockchain-based games can be programmed with tokenized digital assets. These digital assets can be traded within a game or externally on the public market. Blockchain technology can also create real-time sports betting with provably fair outcomes.



The main **revenue stream** of MetaZee’s earnings would be generated by commissions – payment (3% - 5%) with a small percentage of consistent winners paying a “premium rate”.

Revenue streams

- It makes its money by taking a commission on any winnings of between 3 and 5 percent
- A small percentage of consistent winners pay a 'premium rate'
- A growing and cash generative business model (which allows for large dividend payments)

To make money our business will be using the software as a Service “XYZ” and will charge a fee for using their API and infrastructure with the help of professional services.

Members of the online community called “miners” maintain it, and when it's linked with other blocks, you have a “Blockchain. Our business has been built on blockchain technology as a reliable means of keeping sensitive data safe. Blockchain's miners earn Zee for their efforts.

Cost centers

Key partners

- Good relationships with many sporting associations and high profile sporting clubs (such as FC Barcelona and Manchester United FC)
- Sport events Organizers
- Gambling locations
- Google Play Store
- Apple Play Store
- Payment and Technology Partners
- Betting and booking companies
- Strategic and Alliance Partners

Key activities

- Develop application in-house +this avoid costs for hiring developers.
- Marketing and advertising
- Online Betting portals maintenance operation.
- Facility Rentals
- Employment
- Host local , national and international events

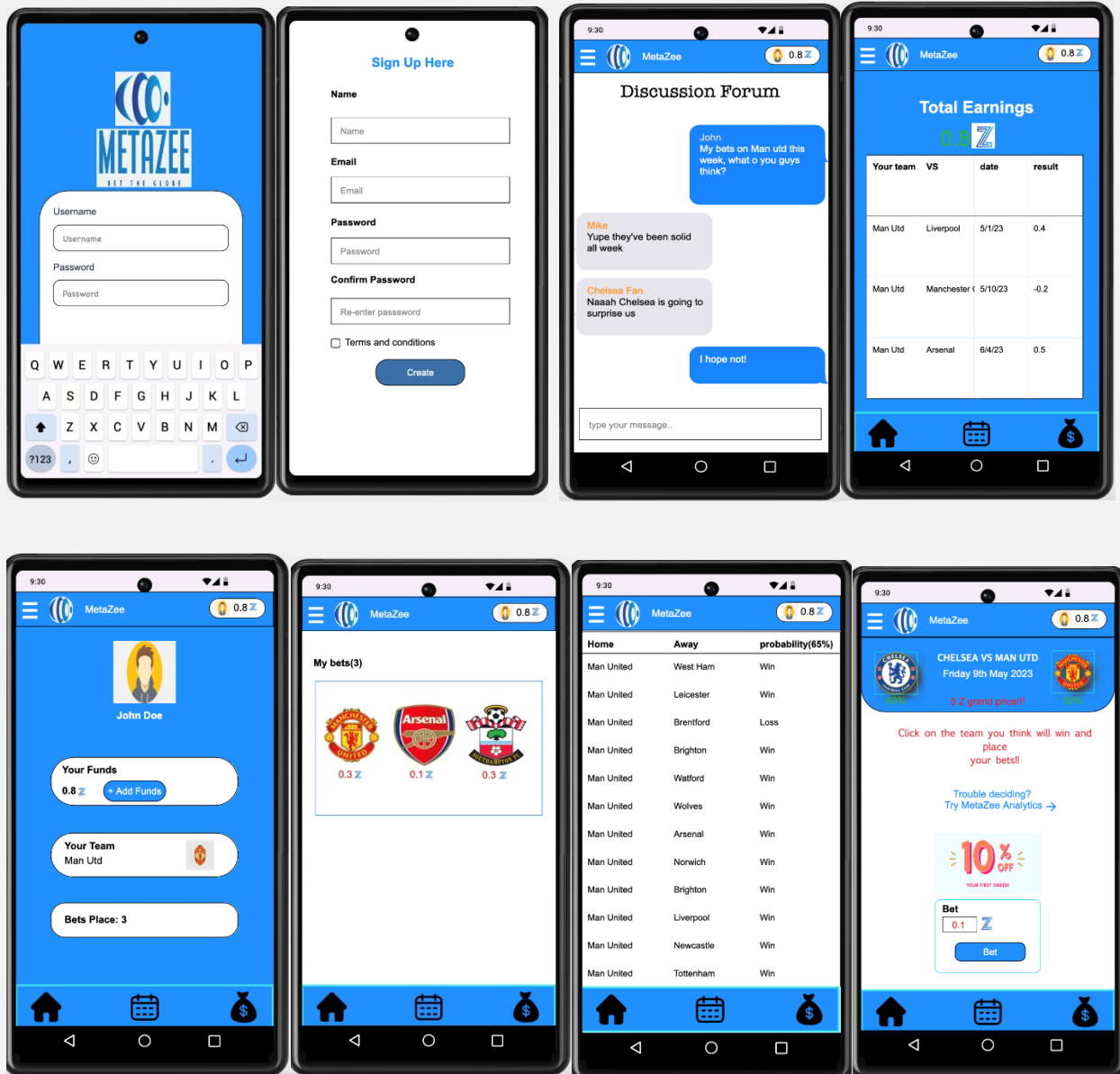
Key resources

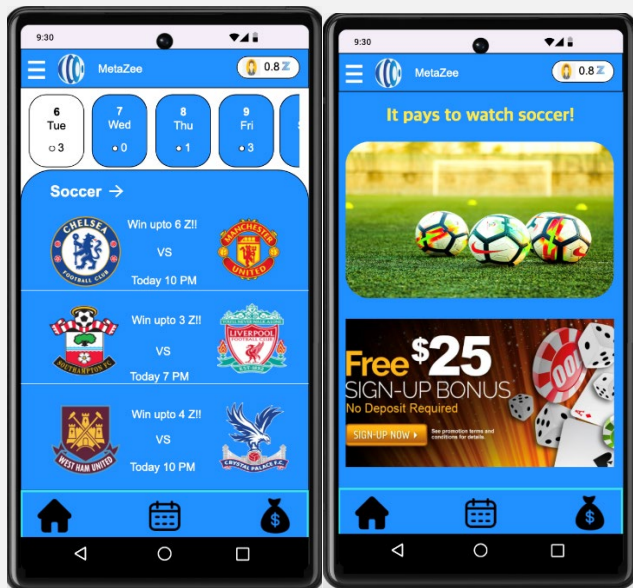
- Reputation and industry standing
- in-house technology
- Patents
- Innovative products
- IT engineers
- Sporting partnership+ games portfolio
- Creative Marketing
- Stock Exchange Model

Cost structures

- The implementation of its marketing and advertising campaigns
- High Fixed costs (investments in facilities ,maintaining costs ,programs and salaries
- High cost of development of its propriety technologies and intellectual properties
- The operation of betting activities
- The procurement of professional services
- Legal
- Operating Licenses and Taxes

Product Prototype





XGBoost Model by Amazon Sagemaker

```

# import libraries
import boto3, re, sys, math, json, os, sagemaker, urllib.request
from sagemaker import get_execution_role
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from IPython.display import Image
from IPython.display import display
from time import gmtime, strftime
from sagemaker.predictor import csv_serializer

# Define IAM role
role = get_execution_role()
prefix = 'sagemaker/DEMO-xgboost-dm'
my_region = boto3.session.Session().region_name # set the region of the instance

# this line automatically looks for the XGBoost image URI and builds an XGBoost container.
xgboost_container = sagemaker.image_uris.retrieve("xgboost", my_region, "latest")

print("Success - the MySageMakerInstance is in the " + my_region + " region. You will use the " + xgboost_container + " container for your SageMaker endpoint.")

Success - the MySageMakerInstance is in the us-west-1 region. You will use the 632365934929.dkr.ecr.us-west-1.amazonaws.com/xgboost:latest container for your SageMaker endpoint.

```

```
[ ] bucket_name = 'predictionenglandleague123' # <--- CHANGE THIS VARIABLE TO A UNIQUE NAME FOR YOUR BUCKET
s3 = boto3.resource('s3')
try:
    if my_region == 'us-east-1':
        s3.create_bucket(Bucket=bucket_name)
    else:
        s3.create_bucket(Bucket=bucket_name, CreateBucketConfiguration={ 'LocationConstraint': my_region })
    print('S3 bucket created successfully')
except Exception as e:
    print('S3 error: ',e)
```

S3 error: An error occurred (BucketAlreadyOwnedByYou) when calling the CreateBucket operation: Your previous request to create the named bucket succeeded and you already own it.

```
[ ] try:
    urllib.request.urlretrieve ("https://drive.google.com/uc?export=download&id=18hvX0380XjjPF6SqM2SM1-C3r54qppxD", "example.csv")
    print('Success: downloaded example.csv.')
except Exception as e:
    print('Data load error: ',e)

try:
    model_data = pd.read_csv('./example.csv', index_col=0)
    print('Success: Data loaded into dataframe.')
except Exception as e:
    print('Data load error: ',e)
```

Success: downloaded example.csv.
Success: Data loaded into dataframe.


```
[ ] train_data, test_data = np.split(model_data.sample(frac=1, random_state=1729), [int(0.7 * len(model_data))])
print(train_data.shape, test_data.shape)
```

(3154, 28) (1352, 28)

```
[ ] pd.concat([train_data['FTR_H'], train_data.drop(['FTR_D', 'FTR_H'], axis=1)], axis=1).to_csv('train.csv', index=False, header=False)
boto3.Session().resource('s3').Bucket(bucket_name).Object(os.path.join(prefix, 'train/train.csv')).upload_file('train.csv')
s3_input_train = sagemaker.inputs.TrainingInput(s3_data='s3://{}/{}/train'.format(bucket_name, prefix), content_type='csv')
```

INFO:boto3.credentials:Found credentials from IAM Role: BaseNotebookInstanceEc2InstanceRole

```
[ ] sess = sagemaker.Session()
xgb = sagemaker.estimator.Estimator(xgboost_container,role, instance_count=1, instance_type='ml.m4.xlarge',output_path='s3://{}/{}/output'.format(bucket_name, prefix),sagemaker_session=sess)
xgb.set_hyperparameters(max_depth=5,eta=0.2,gamma=4,min_child_weight=6,subsample=0.8,silent=0,objective='binary:logistic',num_round=100)
```

 xgb.fit({'train': s3_input_train})

```
INFO:sagemaker:Creating training-job with name: xgboost-2023-05-01-06-34-29-749
2023-05-01 06:34:29 Starting - Starting the training job...
2023-05-01 06:34:55 Starting - Preparing the instances for training.....
2023-05-01 06:35:49 Downloading - Downloading input data...
2023-05-01 06:36:19 Training - Downloading the training image...
2023-05-01 06:36:54 Training - Training image download completed. Training in progress...Arguments: train
[2023-05-01:06:37:05:INFO] Running standalone xgboost training.
[2023-05-01:06:37:05:INFO] Path /opt/ml/input/data/validation does not exist!
[2023-05-01:06:37:05:INFO] File size need to be processed in the node: 0.24mb. Available memory size in the node: 8587.41mb
[2023-05-01:06:37:05:INFO] Determined delimiter of CSV input is ','
[06:37:05] S3DistributionType set as FullyReplicated
[06:37:05] 3154x26 matrix with 82004 entries loaded from /opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28 extra nodes, 8 pruned nodes, max_depth=5
[0]#011train-error:0.168358
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32 extra nodes, 6 pruned nodes, max_depth=5
[1]#011train-error:0.161699
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 36 extra nodes, 2 pruned nodes, max_depth=5
[2]#011train-error:0.158529
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 8 pruned nodes, max_depth=5
[3]#011train-error:0.159163
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32 extra nodes, 8 pruned nodes, max_depth=5
[4]#011train-error:0.1487
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28 extra nodes, 8 pruned nodes, max_depth=5
[5]#011train-error:0.1487
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30 extra nodes, 10 pruned nodes, max_depth=5
[6]#011train-error:0.149334
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28 extra nodes, 14 pruned nodes, max_depth=5
[7]#011train-error:0.148066
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28 extra nodes, 6 pruned nodes, max_depth=5
[8]#011train-error:0.146798
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20 extra nodes, 14 pruned nodes, max_depth=5
[9]#011train-error:0.145847
[06:37:05] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28 extra nodes, 8 pruned nodes, max_depth=5
```

```
[ ] xgb_predictor = xgb.deploy(initial_instance_count=1,instance_type='ml.m4.xlarge')
```

```
INFO:sagemaker:Creating model with name: xgboost-2023-05-01-06-40-24-482
INFO:sagemaker:Creating endpoint-config with name xgboost-2023-05-01-06-40-24-482
INFO:sagemaker:Creating endpoint with name xgboost-2023-05-01-06-40-24-482
-----!
```

```
[ ] from sagemaker.serializers import CSVSerializer
```

```
test_data_array = test_data.drop(['FTR_D', 'FTR_H'], axis=1).values #load the data into an array
xgb_predictor.serializer = CSVSerializer() # set the serializer type
predictions = xgb_predictor.predict(test_data_array).decode('utf-8') # predict!
predictions_array = np.fromstring(predictions[1:], sep=',') # and turn the prediction into an array
print(predictions_array.shape)
```

```
(1352,)
```

```
[ ] cm = pd.crosstab(index=test_data['FTR_H'], columns=np.round(predictions_array), rownames=['Observed'], colnames=['Predicted'])
tn = cm.iloc[0,0]; fn = cm.iloc[1,0]; tp = cm.iloc[1,1]; fp = cm.iloc[0,1]; p = (tp+tn)/(tp+tn+fp+fn)*100
print("\n{0:<20}{1:<4.1f}%\n".format("Overall Classification Rate: ", p))
print("{0:<15}{1:<15}{2:>8}".format("Predicted", "win", "not win"))
print("Observed")
print("{0:<15}{1:<2.0f}% ({2:<}){3:>6.0f}% ({4:<})".format("win", tn/(tn+fn)*100,tn, fp/(tp+fp)*100, fp))
print("{0:<16}{1:<1.0f}% ({2:<}){3:>7.0f}% ({4:<}) \n".format("not win", fn/(tn+fn)*100,fn, tp/(tp+fp)*100, tp))
```

```
Overall Classification Rate: 84.1%
```

Predicted	win	not win
Observed		
win	85% (637)	17% (100)
not win	15% (115)	83% (500)

```
[ ] test_data['predictions'] = predictions
```

```
[ ] test_df['prediction'] = result_df.values
```

```
[ ] test_data.to_excel('test_data.xlsx', index=False)
```