*Ex. Age - Replaced invalid values with the mean \*\*\**Input your variables and cleaning

Income/Expenditure/Gifttransaction/Transactionamount:

1.create a new variable to capture the currency type (AUD, AED, GBP).

2.For each value, if it is in AED or GBP, we will convert it to AUD based on an assumed conversion rate.

3.remove currency symbols like "AU$", "£", and "AED" and keep the numerical values.

Not changed:

* UserID
* Terrorism
* Email
* Occupation
* EducationLevel
* MaritalStatus
* NumDependants
* MerchantID
* IsFraud
* TransactionNumber
* UserTenure
* Latitude
* Longitude
* Transaction Type

Altered:

Age - ages were absolute valued to account for incorrectly inputted values that were negative. Values over 1000 were divided by 1000 as extra zeros were an error.

Income:

GiftsTransaction

TransactionDate :- checked for missing values on excel using the =ISBLANK() function.

Transaction Time :- checked for the missing values on excel using the =ISBLANK() function. Some values in the transaction time column did not make sense or follow a uniform format. So I substituted the “/” with “:” to get all the values in a sensible format.

Transaction Amount

Transaction Location - abbreviations and spelling errors were changed to major cities full name

Expenditure

DeviceType - condensed ie. android or iphone = mobile

YC Notes \*\*\* Don’t edit

Grid Search

Classification report

Ridge and logistic regression

Time and

import pandas as pd

import xgboost as xgb

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report, log\_loss, roc\_auc\_score, f1\_score

def xgboost\_evaluation(x\_train, y\_train, x\_test, test\_size=0.2, random\_state=42):

# Separate the TransactionNumber for later use

transaction\_numbers = x\_test['TransactionNumber']

# Drop TransactionNumber for training and prediction

x\_train\_clean = x\_train.drop(columns=['TransactionNumber'], errors='ignore')

x\_test\_clean = x\_test.drop(columns=['TransactionNumber'], errors='ignore')

# Split the training data into training and validation sets

x\_train\_split, x\_val\_split, y\_train\_split, y\_val\_split = train\_test\_split(

x\_train\_clean, y\_train, test\_size=test\_size, random\_state=random\_state

)

# Convert y\_train\_split and y\_val\_split to 1D if they are DataFrames

if isinstance(y\_train\_split, pd.DataFrame):

y\_train\_split = y\_train\_split.iloc[:, 0]

if isinstance(y\_val\_split, pd.DataFrame):

y\_val\_split = y\_val\_split.iloc[:, 0]

# Create the XGBoost model with more estimators

model = xgb.XGBClassifier(

objective='binary:logistic',

eval\_metric='logloss',

use\_label\_encoder=False,

n\_estimators=1000, # Increased number of estimators

max\_depth=10,

learning\_rate=0.1,

random\_state=random\_state

)

# Fit the model on the training data with early stopping

model.fit(

x\_train\_split, y\_train\_split.values.ravel(), # Ensure y\_train\_split is 1D

eval\_set=[(x\_val\_split, y\_val\_split)],

early\_stopping\_rounds=50, # Stop if no improvement for 50 rounds

verbose=True

)

# Evaluate the model on the validation set

y\_val\_pred = model.predict(x\_val\_split)

y\_val\_pred\_proba = model.predict\_proba(x\_val\_split)[:, 1]

auc = roc\_auc\_score(y\_val\_split, y\_val\_pred\_proba)

f1 = f1\_score(y\_val\_split, y\_val\_pred, average='weighted')

ll = log\_loss(y\_val\_split, y\_val\_pred\_proba)

print(f"Validation Log Loss: {ll:.4f}")

print(f"Validation AUC: {auc:.4f}")

print(f"Validation F1 Score: {f1:.4f}")

# Generate predictions on the test data

y\_test\_pred = model.predict(x\_test\_clean)

# Include TransactionNumber in the output and ensure it is an integer

y\_test\_pred\_df = pd.DataFrame({

'TransactionNumber': transaction\_numbers.astype(int), # Convert to integer

'IsFraud': y\_test\_pred

})

# Save predictions to a CSV file

y\_test\_pred\_df.to\_csv('xgboost\_predictions.csv', index=False)

return y\_test\_pred\_df

# Example usage

# x\_train, y\_train, and x\_test should be defined and preprocessed earlier in your code

# Perform the evaluation and save results

y\_test\_pred\_df = xgboost\_evaluation(x\_train, y\_train, x\_test)

# Print or use the predictions

print("Test Predictions:\n", y\_test\_pred\_df.head())

| @outlook.com |
| --- |
| @gmail.com |
| @yahoo.com |
| @disposable.com |
| @securemail.com |
| @tempmail.com |

| @disposable.com |
| --- |
| @gmail.com |
| @tempmail.com |
| @outlook.com |
| @yahoo.com |
| @securemail.com |

EDAs

Relationship

No relationship