```
from google.colab import files
uploaded = files.upload()
```

```
<IPython.core.display.HTML object>
```

```
Saving FINAL_USO.csv to FINAL_USO (2).csv
```

```
print (uploaded['FINAL_USO.csv'][:200].decode('utf-8') + '...')
```

```
Date,Open,High,Low,Close,Adj 
Close,Volume,SP_open,SP_high,SP_low,SP_close,SP_Ajclose,SP_volume,DJ_o
pen,DJ_high,DJ_low,DJ_close,DJ_Ajclose,DJ_volume,EG_open,EG_high,EG_lo
w,EG_close,EG_Ajclose,EG_volume...
```

```
import pandas as pd
import io

df = pd.read_csv(io.StringIO(uploaded['FINAL_USO.csv'].decode('utf-
8')))
df
```

```
            Date        Open        High         Low       Close   Adj 
Close  \
0     2011-12-15  154.740005  154.949997  151.710007  152.330002   
152.330002   
1     2011-12-16  154.309998  155.369995  153.899994  155.229996   
155.229996   
2     2011-12-19  155.479996  155.860001  154.360001  154.869995   
154.869995   
3     2011-12-20  156.820007  157.429993  156.580002  156.979996   
156.979996   
4     2011-12-21  156.979996  157.529999  156.130005  157.160004   
157.160004   
...          ...         ...         ...         ...         ...   
...   
1713  2018-12-24  119.570000  120.139999  119.570000  120.019997   
120.019997   
1714  2018-12-26  120.620003  121.000000  119.570000  119.660004   
119.660004   
1715  2018-12-27  120.570000  120.900002  120.139999  120.570000   
120.570000   
1716  2018-12-28  120.800003  121.080002  120.720001  121.059998   
121.059998   
1717  2018-12-31  120.980003  121.260002  120.830002  121.250000   
121.250000   

        Volume     SP_open     SP_high      SP_low  ...    GDX_Low  
GDX_Close  \
0     21521900  123.029999  123.199997  121.989998  ...  51.570000   
51.680000   
1     18124300  122.230003  122.949997  121.300003  ...  52.040001   
```

```
52.680000
2     12547200  122.059998  122.320000  120.029999  ...  51.029999
51.169998
3      9136300  122.180000  124.139999  120.370003  ...  52.369999
52.990002
4     11996100  123.930000  124.360001  122.750000  ...  52.419998
52.959999
...        ...         ...         ...         ...  ...        ...
...
1713   9736400  239.039993  240.839996  234.270004  ...  20.650000
21.090000
1714  14293500  235.970001  246.179993  233.759995  ...  20.530001
20.620001
1715  11874400  242.570007  248.289993  238.960007  ...  20.700001
20.969999
1716   6864700  249.580002  251.399994  246.449997  ...  20.570000
20.600000
1717   8449400  249.559998  250.190002  247.470001  ...  20.559999
21.090000

      GDX_Adj Close  GDX_Volume   USO_Open   USO_High    USO_Low
USO_Close  \
0         48.973877    20605600  36.900002  36.939999  36.049999
36.130001
1         49.921513    16285400  36.180000  36.500000  35.730000
36.270000
2         48.490578    15120200  36.389999  36.450001  35.930000
36.200001
3         50.215282    11644900  37.299999  37.610001  37.220001
37.560001
4         50.186852     8724300  37.669998  38.240002  37.520000
38.110001
...             ...         ...        ...        ...        ...
...
1713      21.090000    60507000   9.490000   9.520000   9.280000
9.290000
1714      20.620001    76365200   9.250000   9.920000   9.230000
9.900000
1715      20.969999    52393000   9.590000   9.650000   9.370000
9.620000
1716      20.600000    49835000   9.540000   9.650000   9.380000
9.530000
1717      21.090000    53866600   9.630000   9.710000   9.440000
9.660000

      USO_Adj Close  USO_Volume
0         36.130001    12616700
1         36.270000    12578800
2         36.200001     7418200
3         37.560001    10041600
```

```
4         38.110001    10728000
...             ...         ...
1713       9.290000    21598200
1714       9.900000    40978800
1715       9.620000    36578700
1716       9.530000    22803400
1717       9.660000    28417400

[1718 rows x 81 columns]
```

```
df1 = df[['Date', 'Close']]

# Print the new dataframe
print(df1)
```

```
            Date       Close
0     2011-12-15  152.330002
1     2011-12-16  155.229996
2     2011-12-19  154.869995
3     2011-12-20  156.979996
4     2011-12-21  157.160004
...          ...         ...
1713  2018-12-24  120.019997
1714  2018-12-26  119.660004
1715  2018-12-27  120.570000
1716  2018-12-28  121.059998
1717  2018-12-31  121.250000

[1718 rows x 2 columns]
```

```
# set the 'Name' column as the index
df1.set_index('Date', inplace=True)
df1
```

```
                 Close
Date
2011-12-15  152.330002
2011-12-16  155.229996
2011-12-19  154.869995
2011-12-20  156.979996
2011-12-21  157.160004
...                ...
2018-12-24  120.019997
2018-12-26  119.660004
2018-12-27  120.570000
2018-12-28  121.059998
2018-12-31  121.250000

[1718 rows x 1 columns]
```

```python
from sklearn.preprocessing import LabelEncoder

# Define the label encoder
label_encoder = LabelEncoder()

# Convert the float column to integer using the label encoder
df1['Close'] = label_encoder.fit_transform(df1['Close'].astype(int))
```

```
<ipython-input-28-7285e4d80280>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df1['Close'] = label_encoder.fit_transform(df1['Close'].astype(int))
```

```python
df1
```

```
            Close
Date
2011-12-15     47
2011-12-16     50
2011-12-19     49
2011-12-20     51
2011-12-21     52
...           ...
2018-12-24     20
2018-12-26     19
2018-12-27     20
2018-12-28     21
2018-12-31     21

[1718 rows x 1 columns]
```

```python
df1 = df1.dropna()
print(df1)
```

```
            Close
Date
2011-12-15     47
2011-12-16     50
2011-12-19     49
2011-12-20     51
2011-12-21     52
...           ...
2018-12-24     20
2018-12-26     19
2018-12-27     20
2018-12-28     21
2018-12-31     21
```

```
[1718 rows x 1 columns]
```

```
df2=df1
df2
```

```
            Close
Date             
2011-12-15     47
2011-12-16     50
2011-12-19     49
2011-12-20     51
2011-12-21     52
...           ...
2018-12-24     20
2018-12-26     19
2018-12-27     20
2018-12-28     21
2018-12-31     21

[1718 rows x 1 columns]
```

```
df1['S_3'] = df1['Close'].rolling(window=3).mean()
df1['S_9'] = df1['Close'].rolling(window=9).mean()
df1['next_day_price'] = df1['Close'].shift(-1)
df1
```

```
            Close        S_3        S_9  next_day_price
Date                                                   
2011-12-15     47        NaN        NaN            50.0
2011-12-16     50        NaN        NaN            49.0
2011-12-19     49  48.666667        NaN            51.0
2011-12-20     51  50.000000        NaN            52.0
2011-12-21     52  50.666667        NaN            51.0
...           ...        ...        ...             ...
2018-12-24     20  19.000000  17.777778            19.0
2018-12-26     19  19.000000  18.000000            20.0
2018-12-27     20  19.666667  18.333333            21.0
2018-12-28     21  20.000000  18.777778            21.0
2018-12-31     21  20.666667  19.222222             NaN

[1718 rows x 4 columns]
```

```
df1 = df1.dropna()
```

```
# Convert the float column to integer using the label encoder
df1['Close'] = label_encoder.fit_transform(df1['Close'].astype(int))

df1['S_3'] = label_encoder.fit_transform(df1['S_3'].astype(int))
df1['S_9'] = label_encoder.fit_transform(df1['S_9'].astype(int))
```

```
df1['next_day_price'] =
label_encoder.fit_transform(df1['next_day_price'].astype(int))
```

```
<ipython-input-34-dee713ed42f2>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df1['Close'] = label_encoder.fit_transform(df1['Close'].astype(int))
<ipython-input-34-dee713ed42f2>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df1['S_3'] = label_encoder.fit_transform(df1['S_3'].astype(int))
<ipython-input-34-dee713ed42f2>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df1['S_9'] = label_encoder.fit_transform(df1['S_9'].astype(int))
<ipython-input-34-dee713ed42f2>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df1['next_day_price'] =
label_encoder.fit_transform(df1['next_day_price'].astype(int))
```

```
df1
```

```
            Close  S_3  S_9  next_day_price
Date
2011-12-28     46   45   48              45
2011-12-29     45   43   48              46
2011-12-30     46   42   47              50
2012-01-03     50   44   48              51
2012-01-04     51   46   48              52
...           ...  ...  ...             ...
2018-12-21     18   17   16              20
2018-12-24     20   18   16              19
2018-12-26     19   18   17              20
```

```
2018-12-27     20   18   17              21
2018-12-28     21   19   17              21

[1709 rows x 4 columns]
```

```python
df1 = df1.dropna()
X = df1[['S_3', 'S_9']]

# Define dependent variable
y = df1['next_day_price']
df1
```

```
            Close  S_3  S_9  next_day_price
Date
2011-12-28     46   45   48              45
2011-12-29     45   43   48              46
2011-12-30     46   42   47              50
2012-01-03     50   44   48              51
2012-01-04     51   46   48              52
...           ...  ...  ...             ...
2018-12-21     18   17   16              20
2018-12-24     20   18   16              19
2018-12-26     19   18   17              20
2018-12-27     20   18   17              21
2018-12-28     21   19   17              21

[1709 rows x 4 columns]
```

```python
from sklearn.model_selection import train_test_split
import numpy as np

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.3,random_state=80)

# Load libraries
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
import pandas as pd
from sklearn.tree import DecisionTreeClassifier # Import Decision Tree
Classifier
from sklearn.model_selection import train_test_split # Import
train_test_split function
from sklearn import metrics #Import scikit-learn metrics module for
accuracy calculation
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
# Train Decision Tree Classifer
```

```
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
print('Error', np.sqrt(mean_squared_error(y_test, y_pred)))
```

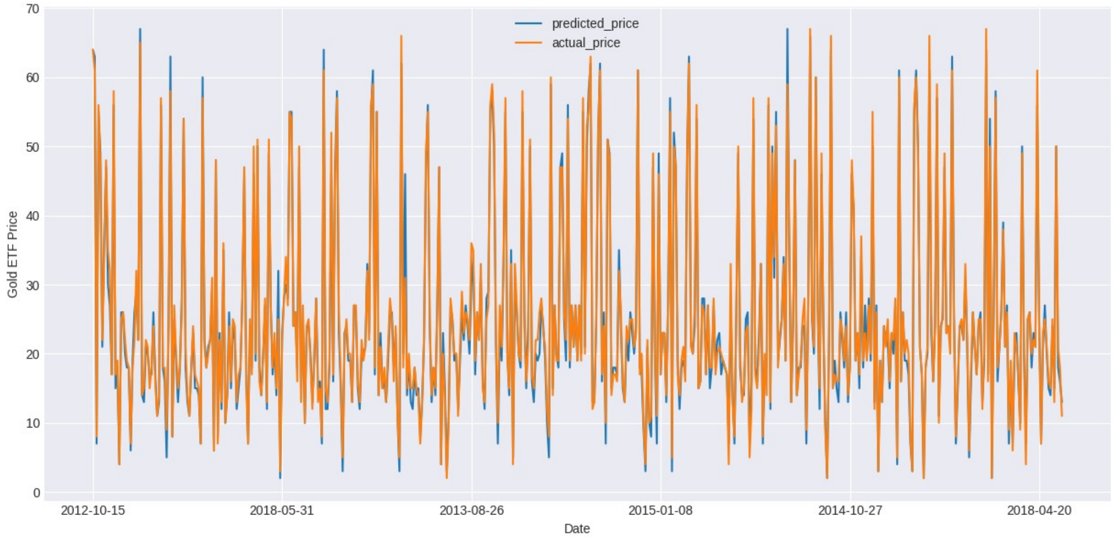
```
Error 1.8995228702237106
```

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.24951267056530213
```

```
import matplotlib.pyplot as plt
%matplotlib inline

plt.style.use('seaborn-darkgrid')
predicted_price = pd.DataFrame(
    y_pred, index=y_test.index, columns=['price'])
predicted_price.plot(figsize=(15, 7))
y_test.plot()
plt.legend(['predicted_price', 'actual_price'])
plt.ylabel("Gold ETF Price")
plt.show()
```

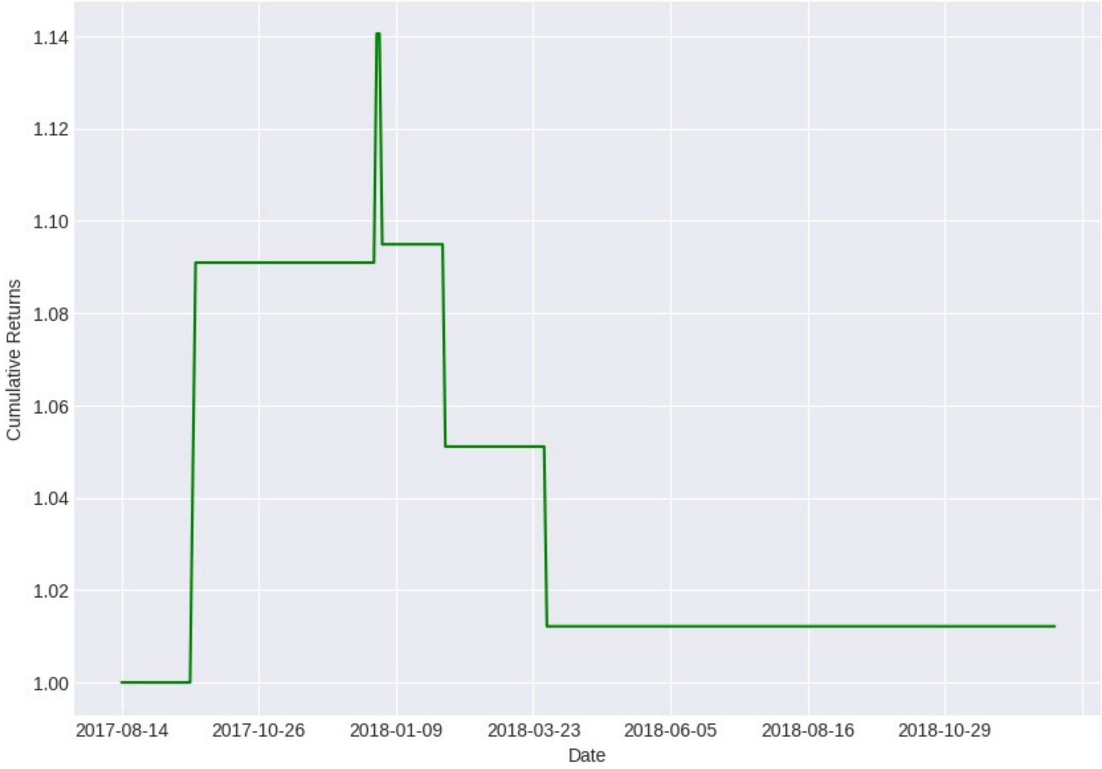
```
<ipython-input-43-5c3e46183941>:4: MatplotlibDeprecationWarning: The
seaborn styles shipped by Matplotlib are deprecated since 3.6, as they
no longer correspond to the styles shipped by seaborn. However, they
will remain available as 'seaborn-v0_8-<style>'. Alternatively,
directly use the seaborn API instead.
  plt.style.use('seaborn-darkgrid')
```



```
r2_score = clf.score(X_test, y_test)*100
float("{0:.2f}".format(r2_score))
```

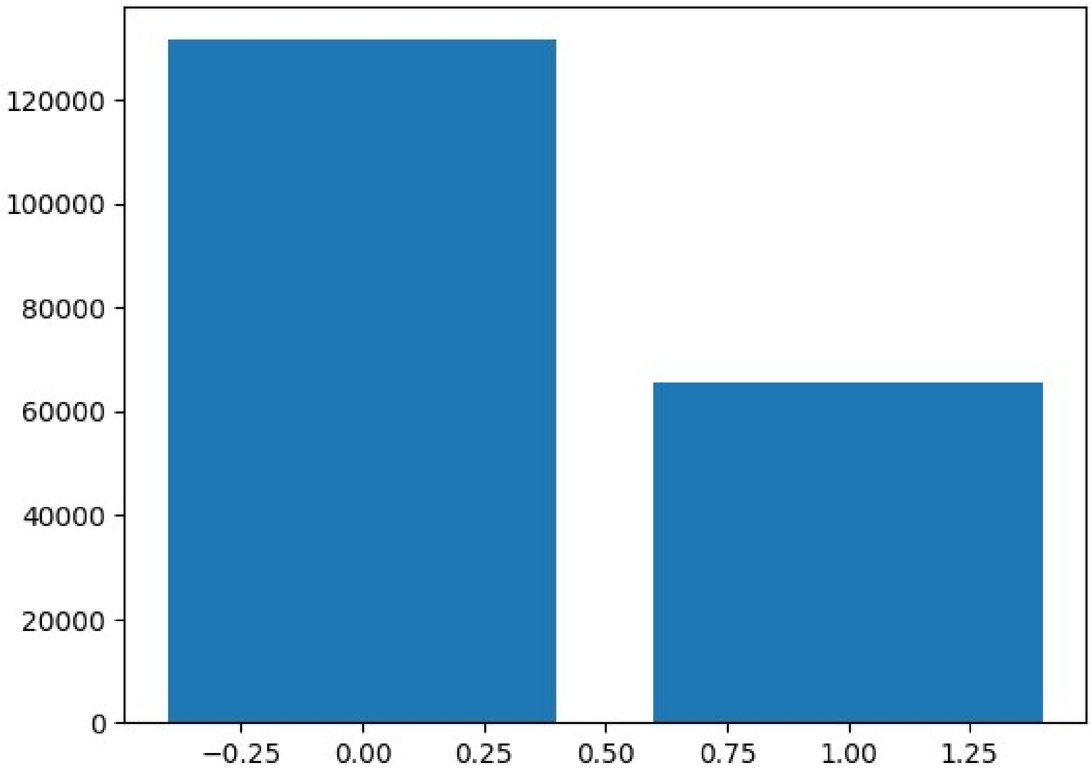
```
24.95
```

```
t = .8
t = int(t*len(df1))

gold = pd.DataFrame()

gold['price'] = df1[t:]['Close']
gold['predicted_price_next_day'] = predicted_price
gold['actual_price_next_day'] = y_test
gold['gold_returns'] = gold['price'].pct_change().shift(-1)

gold['signal'] = np.where(gold.predicted_price_next_day.shift(1) <
gold.predicted_price_next_day,1,0)

gold['strategy_returns'] = gold.signal * gold['gold_returns']
((gold['strategy_returns']
+1).cumprod()).plot(figsize=(10,7),color='g')
plt.ylabel('Cumulative Returns')
plt.show()
```



```
# feature selection
def select_features(X_train, y_train, X_test):
    # configure to select all features
    fs = SelectKBest(score_func=f_regression, k='all')
    # learn relationship from training data
    fs.fit(X_train, y_train)
    # transform train input data
    X_train_fs = fs.transform(X_train)
    # transform test input data
    X_test_fs = fs.transform(X_test)
    return X_train_fs, X_test_fs, fs
    # feature selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_regression
X_train_fs, X_test_fs, fs = select_features(X_train, y_train, X_test)

# what are scores for the features
from matplotlib import pyplot
for i in range(len(fs.scores_)):
    print('Feature %d: %f' % (i, fs.scores_[i]))
# plot the scores
pyplot.bar([i for i in range(len(fs.scores_))], fs.scores_)
pyplot.show()
```

```
Feature 0: 131497.280262
Feature 1: 65325.270672
```



```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
```

```
[[3 1 0 ... 0 0 0]
 [1 2 0 ... 0 0 0]
 [0 1 2 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 1]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 1 0]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           2       0.75      0.75      0.75         4
           3       0.29      0.67      0.40         3
           4       0.67      0.33      0.44         6
           5       0.00      0.00      0.00         5
           6       0.50      0.33      0.40         3
           7       0.17      0.50      0.25         6
           8       0.25      0.12      0.17         8
           9       0.00      0.00      0.00         5
          10       0.50      0.38      0.43         8
          11       0.60      0.43      0.50         7
          12       0.00      0.00      0.00         6
          13       0.32      0.35      0.33        20
          14       0.33      0.33      0.33        15
          15       0.12      0.14      0.13        22
          16       0.19      0.14      0.16        22
          17       0.33      0.26      0.29        23
          18       0.32      0.50      0.39        24
          19       0.17      0.22      0.19        23
          20       0.20      0.20      0.20        25
          21       0.44      0.24      0.31        29
          22       0.29      0.20      0.24        25
          23       0.24      0.29      0.26        17
          24       0.27      0.39      0.32        18
          25       0.45      0.19      0.26        27
          26       0.28      0.33      0.30        21
          27       0.36      0.40      0.38        20
          28       0.36      0.36      0.36        11
          29       0.00      0.00      0.00         1
          30       0.00      0.00      0.00         0
          31       0.00      0.00      0.00         2
          32       0.00      0.00      0.00         4
          33       0.33      0.14      0.20         7
          34       0.00      0.00      0.00         4
          35       0.00      0.00      0.00         2
          36       0.00      0.00      0.00         3
          37       0.00      0.00      0.00         1
          38       0.00      0.00      0.00         1
          39       0.00      0.00      0.00         0
          41       1.00      1.00      1.00         1
          44       0.00      0.00      0.00         1
          46       0.00      0.00      0.00         1
          47       0.33      0.50      0.40         6
          48       1.00      0.33      0.50         6
          49       0.00      0.00      0.00         7
          50       0.11      0.17      0.13         6
          51       1.00      0.25      0.40         4
          52       0.50      0.33      0.40         3
          53       0.00      0.00      0.00         2
          54       0.25      0.33      0.29         3
          55       0.18      0.33      0.24         6
          56       0.00      0.00      0.00         5
          57       0.00      0.00      0.00         8
          58       0.00      0.00      0.00         3
          59       0.00      0.00      0.00         4
          60       0.50      0.25      0.33         4
          61       0.20      0.17      0.18         6
          62       0.00      0.00      0.00         1
          63       0.00      0.00      0.00         1
          64       0.25      1.00      0.40         1
          65       0.00      0.00      0.00         1
          66       0.00      0.00      0.00         3
          67       0.00      0.00      0.00         2

    accuracy                           0.25       513
   macro avg       0.23      0.21      0.20       513
weighted avg       0.28      0.25      0.25       513


/usr/local/lib/python3.10/dist-packages/sklearn/metrics/
_classification.py:1344: UndefinedMetricWarning: Precision and F-score
are ill-defined and being set to 0.0 in labels with no predicted
samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Precision and F-score are ill-
defined and being set to 0.0 in labels with no predicted samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classificatio
n.py:1344: UndefinedMetricWarning: Recall and F-score are ill-defined
and being set to 0.0 in labels with no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```