premise: study underlying model dynamics

challenge assumptions on sell-factor causality in prices

install critical libraries to the underlying os

we are installing altair visual libraries which helps visualize statistics from tables into charts and histograms and helps with calculating models.

```
!pip3 install altair
!pip3 install altair-viewer
!pip3 install -U altair_viewer
!pip3 install statsmodels
!pip3 install imblearn
```

```
Requirement already satisfied: altair in /usr/local/lib/python3.10/dist-packages (4.2.2)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair) (4.19.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from altair) (1.23.5)
Requirement already satisfied: pandas>=0.18 in /usr/local/lib/python3.10/dist-packages (from altair) (1.5.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair) (0.12.0)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (0.32.
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (0.13.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair) (2.8.2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair) (2023.3.post1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair) (2.1.3)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.18->alta
Collecting altair-viewer
  Downloading altair_viewer-0.4.0-py3-none-any.whl (844 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 844.5/844.5 kB 6.1 MB/s eta 0:00:00
Requirement already satisfied: altair in /usr/local/lib/python3.10/dist-packages (from altair-viewer) (4.2.2)
Collecting altair-data-server>=0.4.0 (from altair-viewer)
  Downloading altair_data_server-0.4.1-py3-none-any.whl (12 kB)
Requirement already satisfied: portpicker in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair-viewer
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair-viewer) (
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (4.19.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (1.23.5)
Requirement already satisfied: pandas>=0.18 in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (1.5.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair->altair-viewer) (0.12.0)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair-viewe
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair_view
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair-viewer) (
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair->altair-viewer) (2.1
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from portpicker->altair-data-server>=0.4.0->altai
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.18->alta
Installing collected packages: altair-data-server, altair-viewer
Successfully installed altair-data-server-0.4.1 altair-viewer-0.4.0
Requirement already satisfied: altair_viewer in /usr/local/lib/python3.10/dist-packages (0.4.0)
Requirement already satisfied: altair in /usr/local/lib/python3.10/dist-packages (from altair_viewer) (4.2.2)
Requirement already satisfied: altair-data-server>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from altair_viewer) (0.4.1)
Requirement already satisfied: portpicker in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair_viewer
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair_viewer) (
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (4.19.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (1.23.5)
Requirement already satisfied: pandas>=0.18 in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (1.5.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair->altair_viewer) (0.12.0)
Requirement already satisfied: attrs>=22.2.0 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair_viewe
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair_view
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair
```

we are importing the sklearn module for data analysis, visualization, and machine learning.

```
import pandas as pd
import altair as alt
import matplotlib.pyplot as plt #graphics --viz
from imblearn.over_sampling import ADASYN #synthetic minority oversampling
from sklearn.neighbors import KNeighborsClassifier #ML
from sklearn.preprocessing import StandardScaler #---
from statsmodels.tsa.api import VAR #granger causality
from statsmodels.tsa.vector_ar.var_model import VARResults, VARResultsWrapper
from sklearn.model_selection import train_test_split #TTS ,ML
from sklearn.metrics import accuracy_score #error analysis
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import classification_report
from scipy import stats
```

retrieve the data...

grab data from gh using the URL https://raw.githubusercontent.com/stefanbund/py3100/main/binary_binned_pipeline.csv and reading the table using pandas reader and assigning the table as mdf.

```
#load up binary binned pipeline
url_m = 'https://raw.githubusercontent.com/stefanbund/py3100/main/binary_binned_pipeline.csv'
mdf = pd.read_csv(url_m) #make a pandas dataframe
mdf  #matrix dataframe
```

	Unnamed: 0	group	time	s_MP	change	type	p_MP	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precu
0	0	2	1.660222e+12	30.00	-5.333889e-04	precursor	29.99	-0.012510	0.000002	
1	1	4	1.660222e+12	29.83	-6.637375e-05	precursor	29.88	0.002322	-0.000039	
2	2	6	1.660222e+12	29.92	-6.345915e-04	precursor	29.91	0.005934	-0.000019	
3	3	8	1.660222e+12	29.90	-5.020193e-04	precursor	29.91	-0.002813	-0.000010	
4	4	10	1.660223e+12	29.91	-1.469841e-03	precursor	29.90	-0.000211	0.000061	
...	...	...	...	...	...	...	...	...	...	
6411	6411	12824	1.699042e+12	12.09	6.835784e-08	precursor	12.09	0.007670	-0.000135	
6412	6412	12826	1.699043e+12	12.10	8.291806e-05	precursor	12.10	0.128049	0.000514	
6413	6413	12828	1.699044e+12	12.10	4.140439e-04	precursor	12.10	-0.005610	-0.000328	
6414	6414	12830	1.699045e+12	12.18	-3.610930e-03	precursor	12.13	-0.003349	0.002099	
6415	6415	12832	1.699046e+12	12.14	-1.646768e-04	precursor	12.15	-0.003861	0.001729	

6416 rows × 39 columns

we index the columns of the table and give them a dytpe of object.

```
mdf.columns
```

```
Index(['Unnamed: 0', 'group', 'time', 's_MP', 'change', 'type', 'p_MP',
       'precursor_buy_cap_pct_change', 'precursor_ask_cap_pct_change',
```

```
       'precursor_bid_vol_pct_change', 'precursor_ask_vol_pct_change',
       'length', 'sum_change', 'max_surge_mp', 'min_surge_mp',
       'max_precursor_mp', 'min_precursor_mp', 'area', 'surge_targets_met_pct',
       'group.1', 'time.1', 's_MP.1', 'change.1', 'type.1', 'p_MP.1',
       'precursor_buy_cap_pct_change.1', 'precursor_ask_cap_pct_change.1',
       'precursor_bid_vol_pct_change.1', 'precursor_ask_vol_pct_change.1',
       'length.1', 'sum_change.1', 'max_surge_mp.1', 'min_surge_mp.1',
       'max_precursor_mp.1', 'min_precursor_mp.1', 'area.1', 'surge_area',
       'surge_targets_met_pct.1', 'label'],
      dtype='object')
```

reliability of label

correct classification

what is the average "1" trade's value? here we are seperating the coconuts from the monkeys with a label that either meets the surge target percentage or not. The ones that meet that .75 success rate are labeled as 1 in the columns of our data table.

```
mdf[mdf['label']==1]['surge_targets_met_pct'].mean() #.74 or above
```

```
1.1650823181204584
```

we are coding here what is a 1 and what is a zero or what are the coconuts and what are the monkeys with monkeys being 0 and coconuts being 1.

```
ones = mdf[mdf['label']==1]  #good trades
zeroes = mdf[mdf['label']==0] #baddies
```

study the underlying dichotomies in your data

all the ones in the table is put into the array and we have 119 ones.

```
ones.shape[0] # finger monkeys
```

```
119
```

All the 0 of the data table is put into an array and we returned with 6297 zeros.

```
zeroes.shape[0]  #evil monkeys
```

```
6297
```

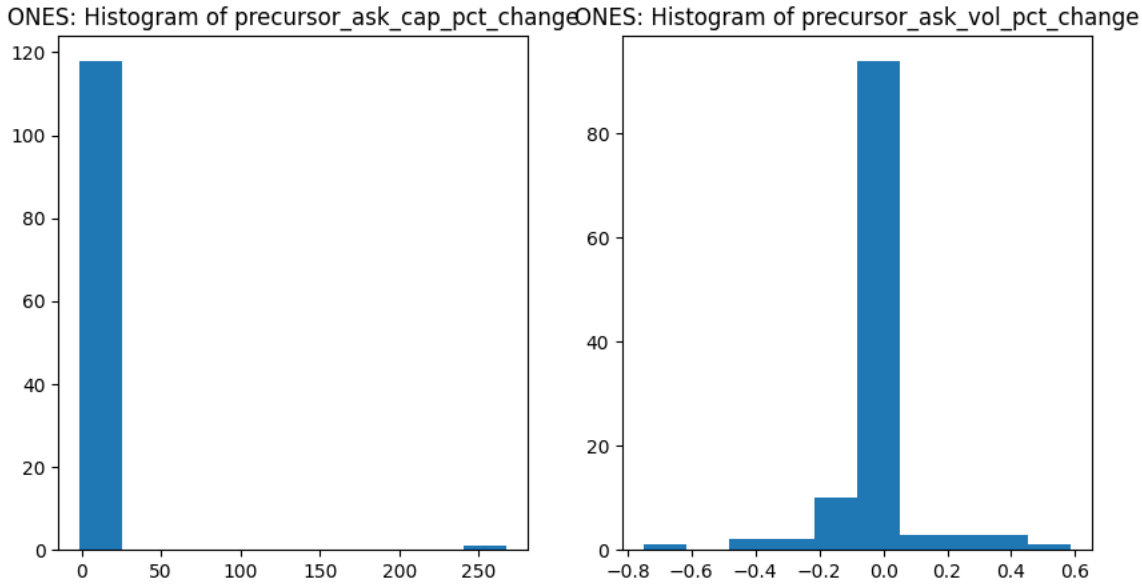
dimensions = features in the data we wish to study, before we classify

we classify what the 2 precursors are to the variable dimensions.

```
dimensions = ['precursor_ask_cap_pct_change','precursor_ask_vol_pct_change']
```

the code below shows our precursors each with how many percent of change and shows them in a histogram named plt. These histograms show the sell orders of the equity and frequency.

```
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].hist(ones['precursor_ask_cap_pct_change'], bins=10)
axs[0].set_title('ONES: Histogram of precursor_ask_cap_pct_change')
axs[1].hist(ones['precursor_ask_vol_pct_change'], bins=10)
axs[1].set_title('ONES: Histogram of precursor_ask_vol_pct_change')
plt.show()
```

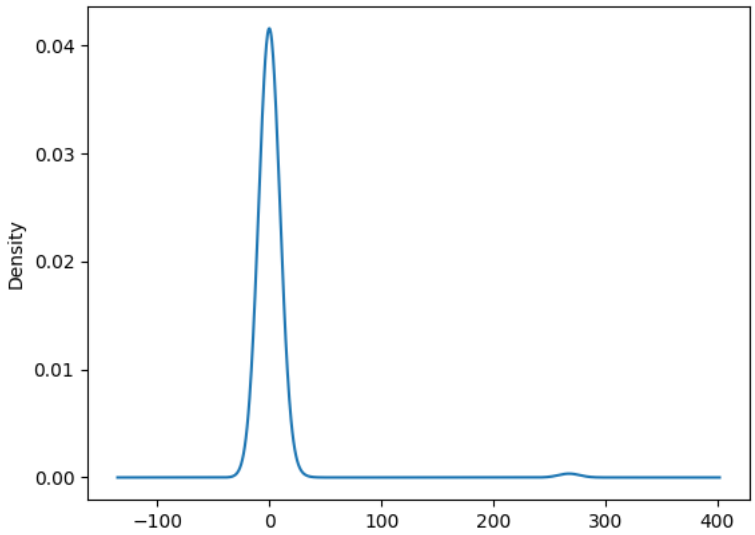


cap vs vol probability density, ONES

we set the density plot of a column called precursor_ask_cap_pct_change, we are using this data to calcuclate the probability density function of the variable of our coconuts.

```
ones['precursor_ask_cap_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

```
<Axes: ylabel='Density'>
```

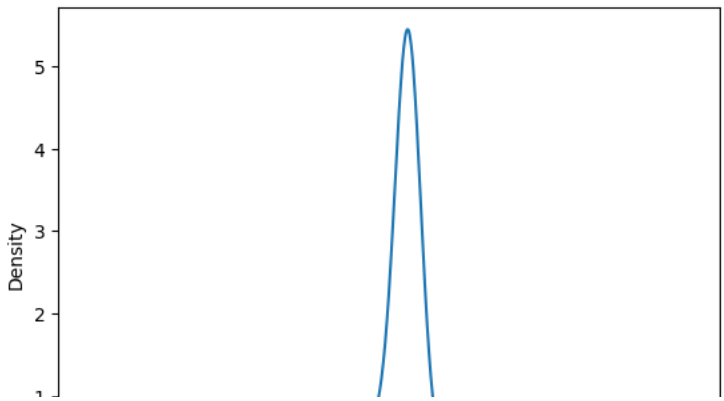


This is calculating probability density function by volume just like above.

this tell us how small or big our margins are to calculater our ones.

```
ones['precursor_ask_vol_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

```
<Axes: ylabel='Density'>
```

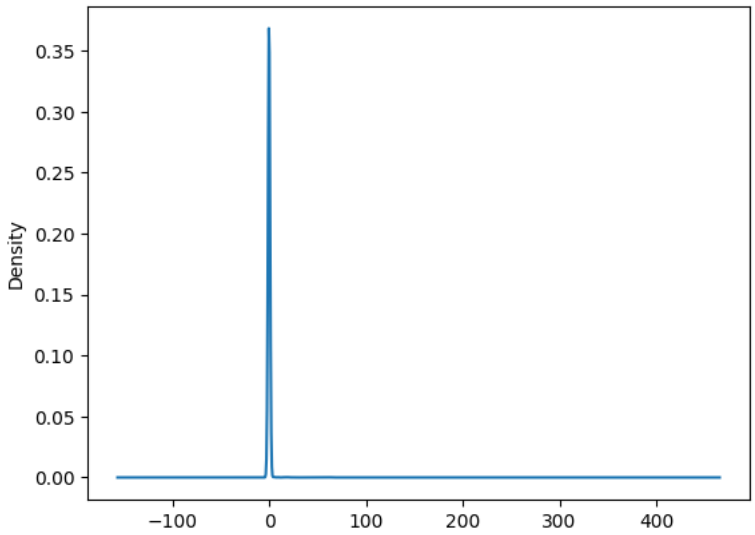


cap vs vol probability density, ZEROES

now we are doing the same with the zeros calculating the probability density function of our zeros

```
zeroes['precursor_ask_cap_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

```
<Axes: ylabel='Density'>
```



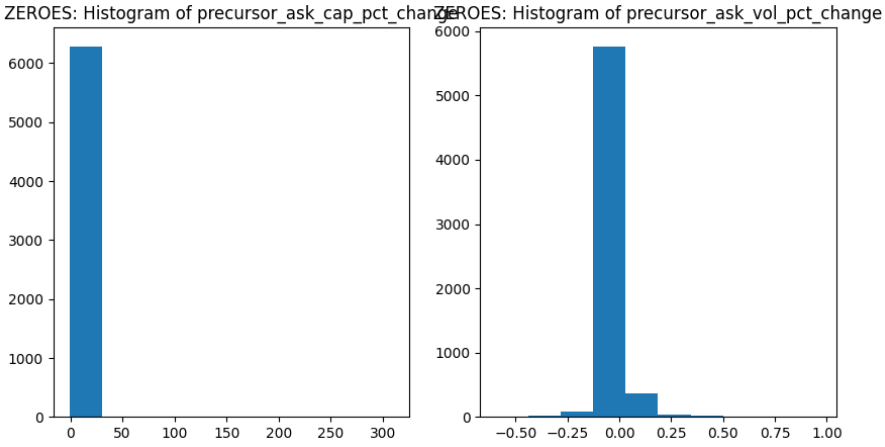
now we are doing the same with the zeros calculating the probability density function with volume of our zeros

```
zeroes['precursor_ask_vol_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

```
<Axes: ylabel='Density'>
```

Here we create the zeros into a histograms with 10 bins of data.

```
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].hist(zeroes['precursor_ask_cap_pct_change'], bins=10)
axs[0].set_title('ZEROES: Histogram of precursor_ask_cap_pct_change')
axs[1].hist(zeroes['precursor_ask_vol_pct_change'], bins=10)
axs[1].set_title('ZEROES: Histogram of precursor_ask_vol_pct_change')
plt.show()
```



Modeling and Prediction Using KNeighbors

we are using KNeighbors classifier algorithm and making a variable name m2_pipeline as our main dataframe and then we define the list of columns used in our dataframe with corr_list and assign the y and x variable with the labels nad keepables of the m2_pipeline dataframe. Hard to understand even with Chat-GPT4 but I think we are trying to extract the labels of our dataframe and use KNeighbors to put them into an array for calculations based on how close or far they are so they can calculate how accurate our zeros and ones are.

```
m2_pipeline = mdf #pd.read_csv("0 Data Processing/binary_binned_pipeline.csv")  #use mdf instead

corr_list = [
'precursor_buy_cap_pct_change', 'precursor_ask_cap_pct_change',
'precursor_bid_vol_pct_change', 'precursor_ask_vol_pct_change','length', 'sum_change', 'surge_targets_met_pct','time', 'label']

m2_pipeline = m2_pipeline[corr_list]
keepable = ['precursor_buy_cap_pct_change',
        'precursor_ask_cap_pct_change',
        'precursor_bid_vol_pct_change',
        'precursor_ask_vol_pct_change',
        'sum_change','length','time']

y = m2_pipeline['label'].values   # y is always a vector, a list of labels
X = m2_pipeline[keepable].values  #x matrix is a list of values/dimensions

X_resampled, y_resampled = ADASYN(random_state=42 ).fit_resample(X, y) #create synthetic classes

X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)

scaler = StandardScaler()  #standardize all numerics
X_train_scaled = scaler.fit_transform(X_resampled)

X_test_scaled = scaler.fit_transform(X_test)

knn = KNeighborsClassifier(algorithm='auto', n_jobs=1, n_neighbors=3)
knn.fit(X_train_scaled, y_resampled)
```

```
KNeighborsClassifier
KNeighborsClassifier(n_jobs=1, n_neighbors=3)
```

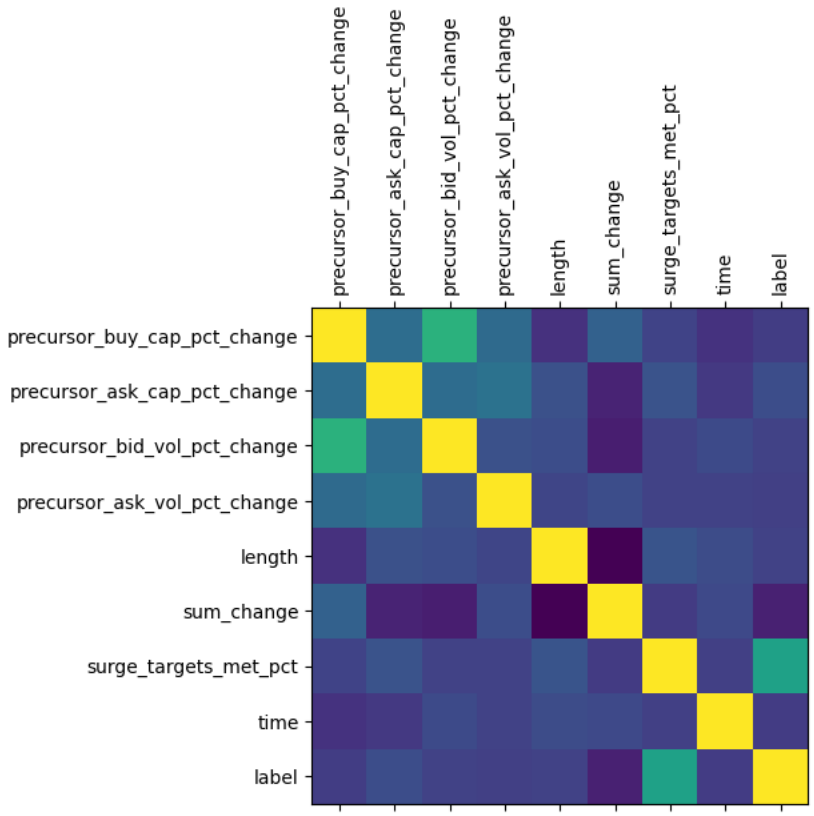
we are setting up the a new matrix table, based on correlation of the precursors.

```
corr_list = [
'precursor_buy_cap_pct_change', 'precursor_ask_cap_pct_change',
'precursor_bid_vol_pct_change', 'precursor_ask_vol_pct_change','length', 'sum_change', 'surge_targets_met_pct','time', 'label']

m2_pipeline = m2_pipeline[corr_list]
m2_pipeline.corr()
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bid_vol_pct_change	precursor_ask_v
precursor_buy_cap_pct_change	1.000000	0.195900	0.547428	
precursor_ask_cap_pct_change	0.195900	1.000000	0.190969	
precursor_bid_vol_pct_change	0.547428	0.190969	1.000000	
precursor_ask_vol_pct_change	0.177817	0.217833	0.058289	
length	-0.074944	0.055215	0.041534	
sum_change	0.136782	-0.131603	-0.151138	
surge_targets_met_pct	-0.001754	0.067987	-0.007312	
time	-0.068998	-0.044788	0.028991	

here we are basically testing and displaying the algorithm we build and displaying it as a heatmap of the correlation.

```
import pandas as pd
import matplotlib.pyplot as plt

# create a sample dataframe
df = m2_pipeline

# calculate the correlation matrix
corr_matrix = df.corr()

# plot the correlation matrix
plt.matshow(corr_matrix)
plt.xticks(range(len(corr_matrix.columns)), corr_matrix.columns, rotation=90)
plt.yticks(range(len(corr_matrix.columns)), corr_matrix.columns)

plt.show()
```



we are using knn classifier to predict our data and ones and zeros.

```
#predict
y_pred_knn = knn.predict(X_test_scaled)
```

```
#plot decision boundary
# Assuming your KNN model is stored in the variable 'knn'
# plot_decision_boundary(knn, X_test_scaled, y_test)
# plt.show()
```

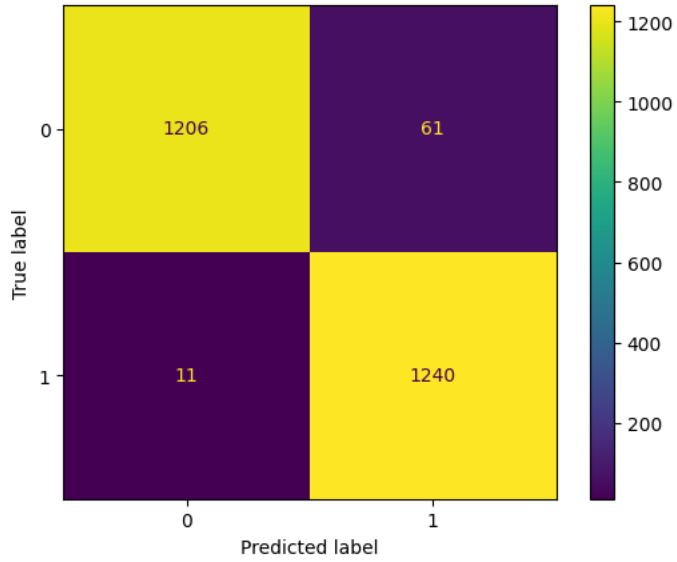
```
# Compute the k-neighborhood graph for your data
graph = knn.kneighbors_graph(X)
graph
```

```
<6416x12589 sparse matrix of type '<class 'numpy.float64'>'
        with 19248 stored elements in Compressed Sparse Row format>
```

- we display of our confusion matrix which shows us how accurate our predictions are with zeros and ones. The average precision came out to 97% out of 1267 for zeros and 1251 for ones.

```
#display confusion matrix

y_pred_knn = knn.predict(X_test_scaled)

accuracy_knn = accuracy_score(y_test, y_pred_knn)
print(accuracy_knn)

labels_ = m2_pipeline['label'].unique()
print(labels_)
ConfusionMatrixDisplay.from_predictions(y_test, y_pred_knn, labels=labels_)

print(classification_report(y_test, y_pred_knn ,zero_division=1))
```

```
0.971405877680699
[0 1]
              precision    recall  f1-score   support

           0       0.99      0.95      0.97      1267
           1       0.95      0.99      0.97      1251

    accuracy                           0.97      2518
   macro avg       0.97      0.97      0.97      2518
weighted avg       0.97      0.97      0.97      2518
```



- Causality Studies

```
ones_r = mdf[mdf['label']==1][keepable]  #good trades
ones_r
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bid_vol_pct
47	0.008169	-0.000036	0
108	-0.002202	-0.000064	-0

we are using the granger causality. Using the table dataframe we determine wether our precusors in and drops are caused by one another in a time series. This helps us determine wether they are related to one another or not.

```
def test_granger(df, p):
    """
    Fits a VAR(p) model on the input df and performs pairwise Granger Causality tests
    """
     # Fit VAR model on first-order differences
    model = VAR(df.diff().dropna())
    results = model.fit(p)
    # Initialize p-value matrix
    p_matrix = pd.DataFrame(index=df.columns, columns=df.columns)
    # Perform pairwise Granger Causality tests
    for caused in df.columns:
        for causing in df.columns:
            if caused != causing:
                test_result = results.test_causality(caused, causing)
                p_value = test_result.pvalue
                p_matrix.loc[caused, causing] = p_value
    # Ensure all columns have float dtypetest_granger
    p_matrix = p_matrix.astype(float)
    return p_matrix

p=7
ones = mdf[mdf['label']==1]  #good trades
p_matrix0 = test_granger(ones_r, p)
caul_mtrx = p_matrix0.rename(index={item: f"{item} caused by" for item in p_matrix0.index})
caul_mtrx.where(caul_mtrx.isna(), caul_mtrx <= 0.01)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarr
  self._init_dates(dates, freq)
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_chang
precursor_buy_cap_pct_change caused by	NaN	Fals
precursor_ask_cap_pct_change caused by	False	Nal
precursor_bid_vol_pct_change caused by	False	Fals
precursor_ask_vol_pct_change caused by	False	Fals

this code shows wether one one unit in the column and one unit with the rows have a causal relationship. NaN means that there isnt enough data, and false means no. We have very few true causality in our time series of the data.

```
zeroes_r = mdf[mdf['label']==0][keepable]  #bad trades
p_matrix1 = test_granger(zeroes_r, p)
caul_mtrx = p_matrix1.rename(index={item: f"{item} caused by" for item in p_matrix1.index})
caul_mtrx.where(caul_mtrx.isna(), caul_mtrx <= 0.01)
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarr
  self._init_dates(dates, freq)
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_chang
precursor_buy_cap_pct_change caused by	NaN	Fals
precursor_ask_cap_pct_change caused by	False	Nal
precursor_bid_vol_pct_change	False	Fals