premise: study underlying model dynamics

challenge assumptions on sell-factor causality in prices

install critical libraries to the underlying os

```
!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>=3.0->altair) (2023
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (0.30.2)
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair) (0.12.0)
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->altair) (1.16.0)
Collecting altair-viewer
  Downloading altair_viewer-0.4.0-py3-none-any.whl (844 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 844.5/844.5 kB 6.6 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) (1.5.2)
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair-viewer) (6.3.2)
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-viewer) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altai
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair-viewer) (0.30.2
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair-viewer) (0.12.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair-viewer) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair-viewer) (2023.3.post1)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair->altair-viewer) (2.1.3)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from portpicker->altair-data-server>=0.4.0->altair-viewer) (5.9.5
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.18->altair->altair-viewe
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) (1.5.2)
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair_viewer) (6.3.2)
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_viewer) (23.1.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altai
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair_viewer) (0.30.2
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->altair_viewer) (0.12.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair_viewer) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair_viewer) (2023.3.post1)
```

```
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

Load a DataFrame named 'mdf' from the provided URL.

```
#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
0	0	2	1.660222e+12	30.00	-5.333889e-04	precursor	29.99	
1	1	4	1.660222e+12	29.83	-6.637375e-05	precursor	29.88	
2	2	6	1.660222e+12	29.92	-6.345915e-04	precursor	29.91	
3	3	8	1.660222e+12	29.90	-5.020193e-04	precursor	29.91	
4	4	10	1.660223e+12	29.91	-1.469841e-03	precursor	29.90	
...	...	...	...	...	...	...	...	
6411	6411	12824	1.699042e+12	12.09	6.835784e-08	precursor	12.09	
6412	6412	12826	1.699043e+12	12.10	8.291806e-05	precursor	12.10	
6413	6413	12828	1.699044e+12	12.10	4.140439e-04	precursor	12.10	
6414	6414	12830	1.699045e+12	12.18	-3.610930e-03	precursor	12.13	
6415	6415	12832	1.699046e+12	12.14	-1.646768e-04	precursor	12.15	

6416 rows × 39 columns

variables associated

Print the columns of Dataframe 'mdf'.

```
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?

Filter rows where the 'label' column is equal to 1 and then calculates the mean of the 'surge_targets_met_pct' column for those filtered row.

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

```
1.1650823181204584
```

Split the DataFrame 'mdf' into two subsets:

'ones': Contains rows where the 'label' is equal to 1 (good trades). 'zeroes': Contains rows where the 'label' is equal to 0 (bad trades).

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

study the underlying dichotomies in your data

Give the number of rows in the 'ones' DataFrame.

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

```
119
```

Give the number of rows in the 'zeroes' DataFrame.

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

```
6297
```

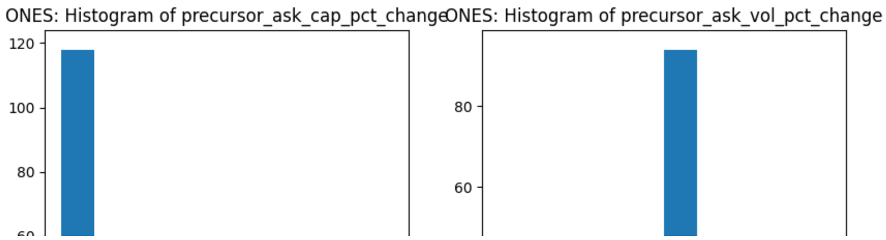
dimensions = features in the data we wish to study, before we classify

A list named 'dimensions' containing two features.

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

Use Matplotlib to create a 1x2 grid of subplots (side by side) with histograms for two columns from the 'ONES' subset of the data, divided into 10 bins each. The resulting figure is displayed.

```
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()
```

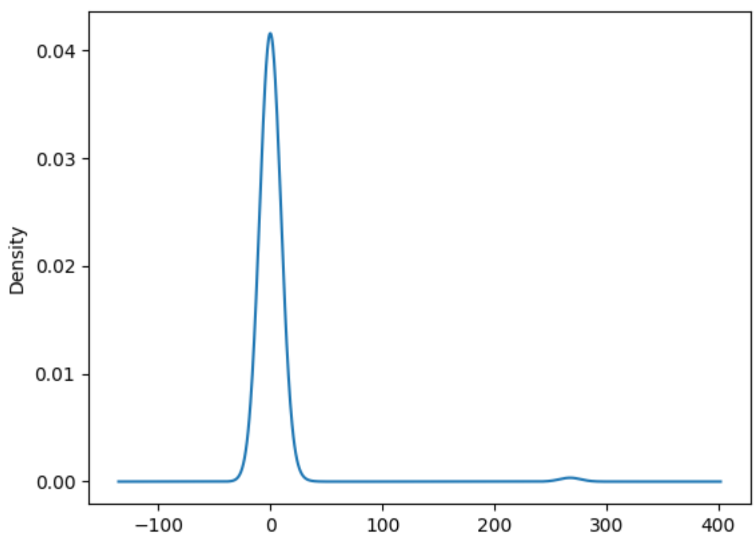


cap vs vol probability density, ONES

Generate a kernel density plot for the column in the 'ONES' subset of the data.

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

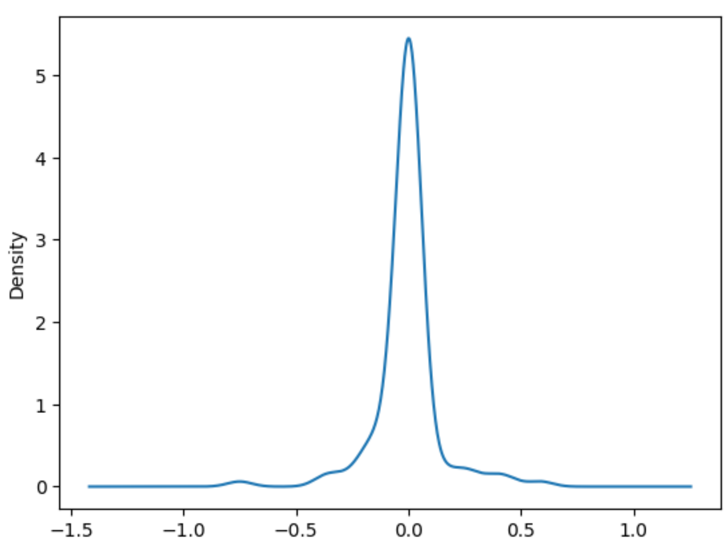
```
<Axes: ylabel='Density'>
```



Repeat the code.

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

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

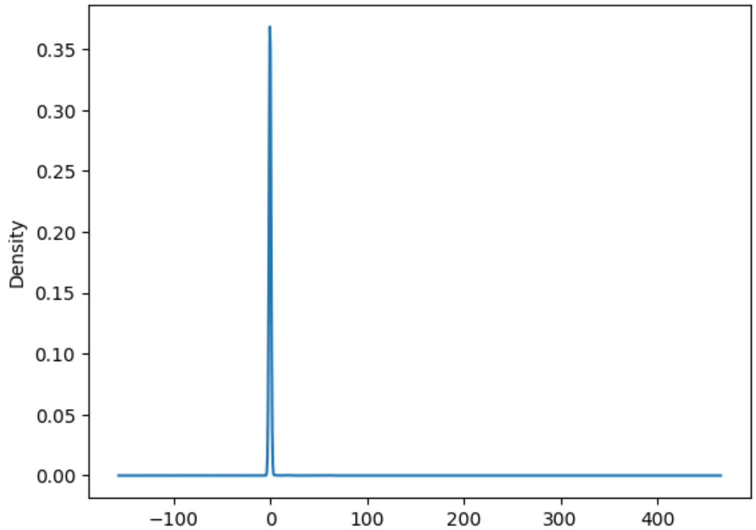


cap vs vol probability density, ZEROES

Generate a kernel density plot for the column in the 'ZEROES' subset of the data.

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

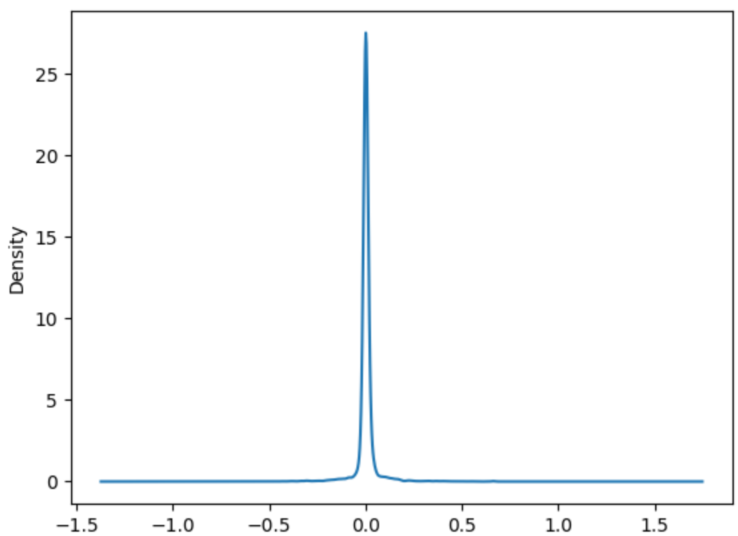
<Axes: ylabel='Density'>



Repeat the code.

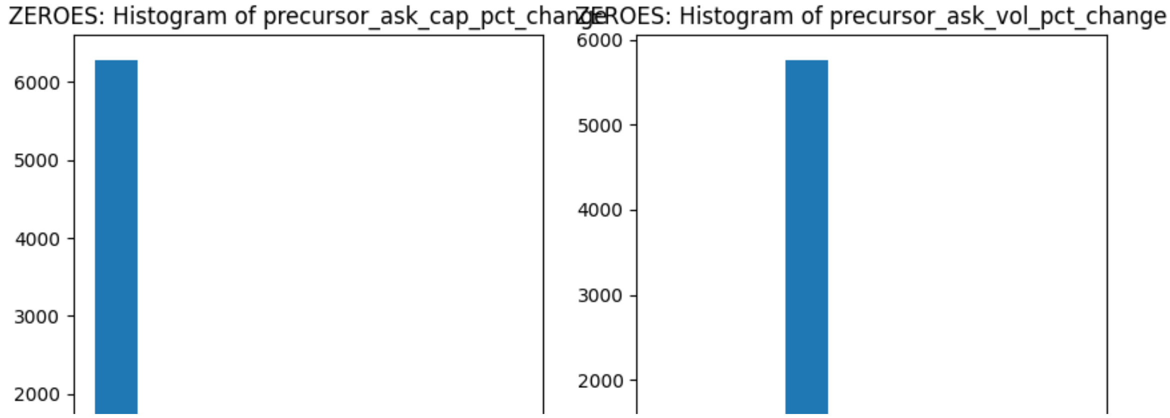
```
zeroes['precursor_ask_vol_pct_change'].plot.density()    #precursor_ask_vol_pct_change
```

<Axes: ylabel='Density'>



Generate two histograms side by side. The left histogram represents the distribution of the 'precursor_ask_cap_pct_change' column in the 'ZEROES' subset, and the right histogram represents the distribution of the 'precursor_ask_vol_pct_change' column in the same subset.

```
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

The code is a machine learning pipeline for classification using the k-nearest neighbors (KNN) algorithm.

The pipeline aims to handle imbalanced classes, split the data, standardize features, and train a KNN classifier for classification tasks.

```
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)
```

Calculate the correlation matrix for selected features in the 'm2_pipeline' DataFrame.

The correlation matrix provides information on the linear relationship between these features. It helps to identify patterns and dependencies among variables in the dataset.

The result is a square matrix where each entry represents the correlation coefficient between two variables. The values range from -1 to 1, indicating the strength and direction of the correlation.

```
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_as
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	
label	-0.025531	0.041780	-0.006628	

Use the 'matshow' function from the 'matplotlib.pyplot' library to create a heatmap of the correlation matrix for the features in the 'm2_pipeline' DataFrame.

The heatmap provides a visual representation of the pairwise correlations between these features. The x-axis and y-axis labels show the feature names, and the color intensity represents the strength and direction 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()
```

Use the trained k-nearest neighbors (KNN) classifier ('knn') to make predictions on the test set ('X_test_scaled'). The predicted labels are stored in the variable 'y_pred_knn'.

```
#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()
```

Create a k-neighbors graph using the 'kneighbors_graph' method of the trained k-nearest neighbors (KNN) classifier ('knn'). The graph represents the connectivity between points in the input data ('X'). The result is stored in the variable 'graph'.

```
# 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>
```

Predicts labels ('y_pred_knn') using the trained k-nearest neighbors (KNN) classifier on the scaled test data ('X_test_scaled'). It then calculates and prints the accuracy of the predictions. The confusion matrix and the classification report are also displayed.

```
#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
```

Causality Studies

Create a DataFrame 'ones_r' containing the rows from 'mdf' where the 'label' is equal to 1, and select only the columns specified in the 'keepable' list.

```
weighted avg       0.97      0.97      0.97      2318
```

```
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_change	precursor_ask_vol_pct_change	sum_(
47	0.008169	-0.000036	0.003968	-0.002134	-0.(
108	-0.002202	-0.000064	-0.003449	-0.011060	-0.2
144	-0.204558	0.000178	-0.088451	0.029849	-0.2
159	0.006487	-0.000134	0.003800	-0.023368	-0.(
189	0.001016	-0.000022	0.002570	-0.002694	-0.3
...	...	...	...	...	
6283	-0.012920	-0.003338	-0.002208	-0.027292	-0.(
6292	-0.008260	0.011126	-0.000741	0.012573	-0.(
6312	0.215995	0.003481	0.027989	0.034112	-0.(
6315	0.059232	0.009487	0.009968	0.036562	-0.(
6395	0.005644	0.002875	0.000943	0.014005	-0.(

119 rows × 7 columns

Define a function 'test_granger' that fits a VAR(p) model on the input DataFrame 'df' and performs pairwise Granger Causality tests. Then, apply this function to the DataFrame 'ones_r' where the 'label' is equal to 1, and create a matrix ('caul_mtrx') of p-values for Granger Causality tests where values less than or equal to 0.01 are replaced with 'True'.

```
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: ValueWarning: An unsupported index was provided and will be ignored wh
  self._init_dates(dates, freq)
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bid_vol_pct_change	precursor_as
precursor_buy_cap_pct_change caused by	NaN	False	False	
precursor_ask_cap_pct_change caused by	False	NaN	False	
precursor_bid_vol_pct_change caused by	False	False	NaN	
precursor_ask_vol_pct_change caused by	False	False	False	

Apply the 'test_granger' function to the DataFrame 'zeroes_r' where the 'label' is equal to 0, and create another matrix ('caul_mtrx') of p-values for Granger Causality tests with a threshold of 0.01.

time caused by	False	False	False

```
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: ValueWarning: An unsupported index was provided and will be ignored wh
  self._init_dates(dates, freq)
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bid_vol_pct_change	precursor_as
precursor_buy_cap_pct_change caused by	NaN	False	False	
precursor_ask_cap_pct_change caused by	False	NaN	True	
precursor_bid_vol_pct_change caused by	False	False	NaN	
precursor_ask_vol_pct_change caused by	False	False	False	
sum_change caused by	False	False	False	
length caused by	False	False	False	
time caused by	True	False	True	