```
## premise: study underlying model dynamics
challenge assumptions on sell-factor causality in prices
```

premise: study underlying model dynamics

challenge assumptions on sell-factor causality in prices

install

Double-click (or enter) to edit

install critical libraries to the underlying os

This code allows us to reach the outer operating system of the notebook using the exclamation point.

```
!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) (2023.7
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->altair-
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-viewer)
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->altair_
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)
```

```
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

This code right here is the data set that has had data excavating and has created this dataset. This data set is the product done from excavating.

```
#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	prec
0	0	2	1.660222e+12	30.00	-5.333889e-04	precursor	29.99	-0.012510	
1	1	4	1.660222e+12	29.83	-6.637375e-05	precursor	29.88	0.002322	
2	2	6	1.660222e+12	29.92	-6.345915e-04	precursor	29.91	0.005934	
3	3	8	1.660222e+12	29.90	-5.020193e-04	precursor	29.91	-0.002813	
4	4	10	1.660223e+12	29.91	-1.469841e-03	precursor	29.90	-0.000211	
...	...	...	...	...	...	...	...	...	
6411	6411	12824	1.699042e+12	12.09	6.835784e-08	precursor	12.09	0.007670	
6412	6412	12826	1.699043e+12	12.10	8.291806e-05	precursor	12.10	0.128049	
6413	6413	12828	1.699044e+12	12.10	4.140439e-04	precursor	12.10	-0.005610	
6414	6414	12830	1.699045e+12	12.18	-3.610930e-03	precursor	12.13	-0.003349	
6415	6415	12832	1.699046e+12	12.14	-1.646768e-04	precursor	12.15	-0.003861	

6416 rows × 39 columns

variables associated

```
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?

This code checks if the pints were profitable or not. The code "']==1]['surge_targets_met_pct'].mean() #.74 or above " means that a label is one if the surge targets are .074 or above. It looks for the average needed.

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

```
1.1650823181204584
```

This creates two data frames, The first ones is the good trades and the second zeroes, the bad. This code basically instructs the machine to be able to distinguish between right and wrong.

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

study the underlying dichotomies in your data

This code tells the machine that ones are the right choice frim evil monkeys, so ones would be correct.

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

```
119
```

Double-click (or enter) to edit

The code here gives a command that zeroes are a bad pick from ones and zeroes.

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

```
6297
```

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

Double-click (or enter) to edit

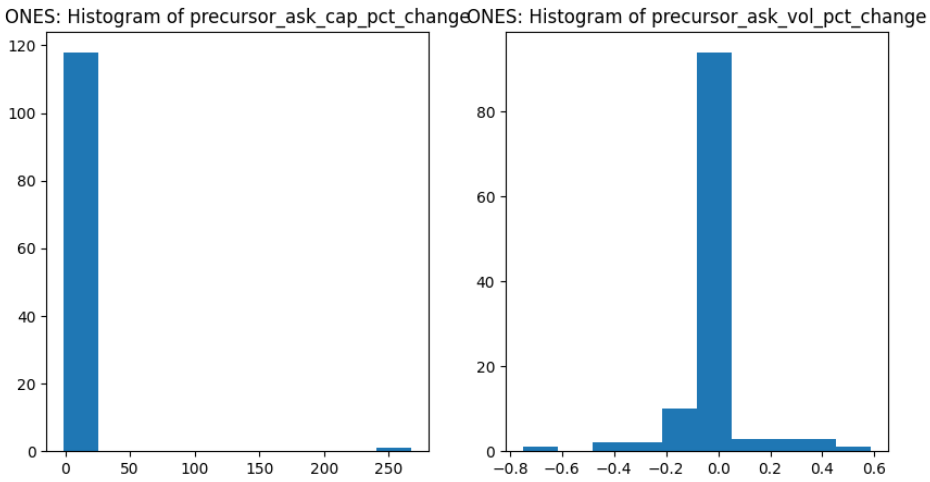
Dimensions are data features that we want to investigate before classifying them. In this code, the "ask_cap" asks the value of all cell orders, and it shows the volume of the numbers and how numerous they are.

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

Double-click (or enter) to edit

This code generates a histogram that allows us to investigate the distribution of capitalization values and volume.

```
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()
```

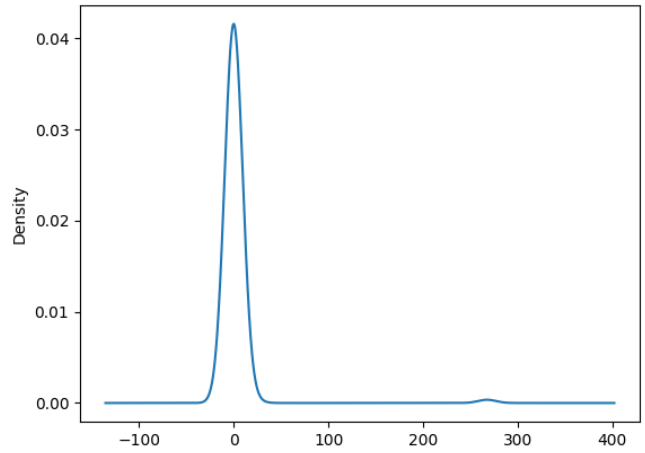


cap vs vol probability density, ONES

this code is the plot of the ones volume change and the ones are marked correct.

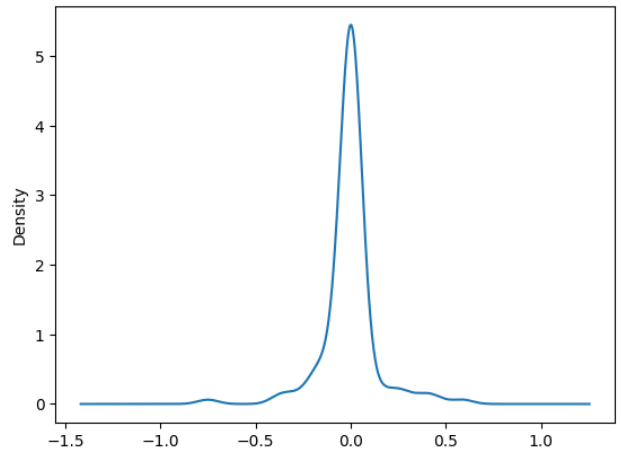
```
ones['precursor_ask_cap_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

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




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

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



cap vs vol probability density, ZEROES

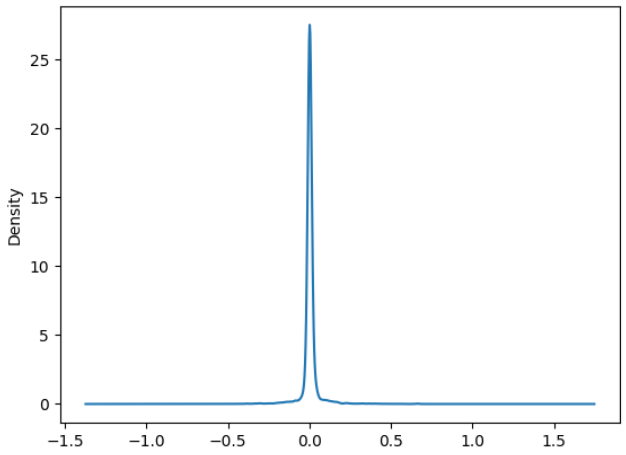
This code would be the plot for the zeros density and volume change.

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

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

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

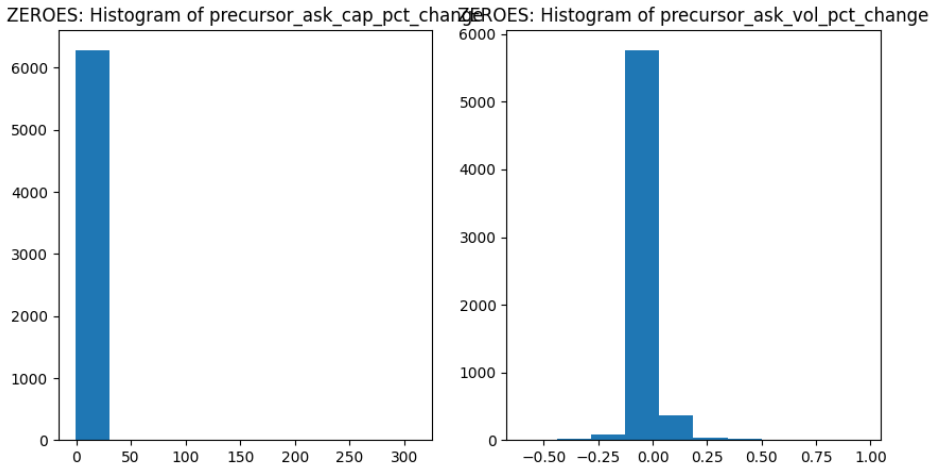
```
<Axes: ylabel='Density'>
```



We're looking at dimensionality in this code, but the variance between instances is very low. It is the zeros that are incorrect and difficult to distinguish. This creates a histogram for us to see.

Double-click (or enter) to edit

```
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

This code includes primary dimensions. Two data sets are involved in the machine learning problem. One consists of an x matrix and a set of values. The other data set is y, a vector that is a list of labels. We provide the model with statistics and features so that they can learn from them.

```
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)
```

In this code we are given the statistics and numbers needed to form a chart from the pervious codes.

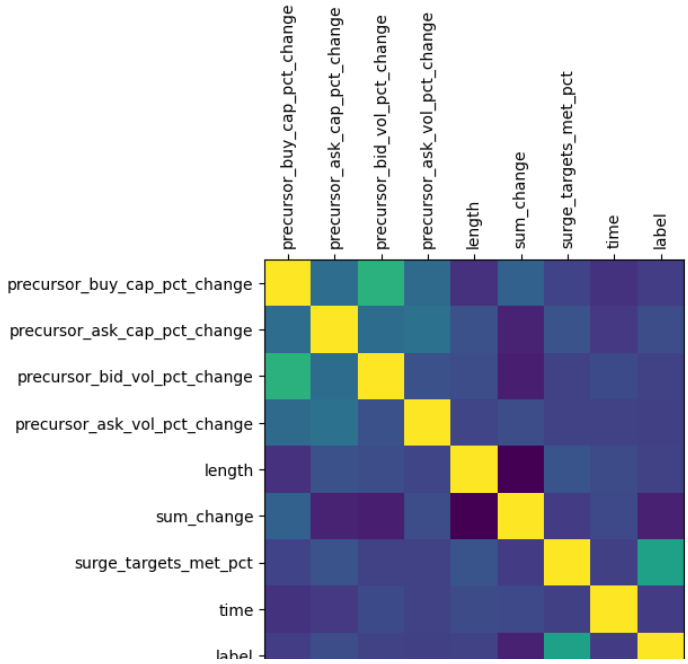
```
corr_list = [
'precursor_buy_cap_pct_change', 'precursor_ask_cap_pct_change',
'precursor_bid_vol_pct_change', 'precursor_ask_vol_pct_change','length', 'sum_change', 'surge_targets_met_pct','time', 'label']

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

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bi
precursor_buy_cap_pct_change	1.000000	0.195900	
precursor_ask_cap_pct_change	0.195900	1.000000	
precursor_bid_vol_pct_change	0.547428	0.190969	
precursor_ask_vol_pct_change	0.177817	0.217833	
length	-0.074944	0.055215	
sum_change	0.136782	-0.131603	
surge_targets_met_pct	-0.001754	0.067987	
time	-0.068998	-0.044788	
label	-0.025531	0.041780	

The code creates a heatmap to display the correlation matrix of the variables in DataFrame m2_pipeline. This is useful for understanding the relationships between the dataset's variables. Stronger correlations are indicated by darker squares, while weaker or no correlations are indicated by lighter squares.

Double-click (or enter) to edit

```
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()
```



Double-click (or enter) to edit

This code helps with the accurate prediction of whats right and whats wrong and weather which one to choose from.

```
#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()
```

This code instructs the model to use the matrix and certain data to formulate a graph.

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

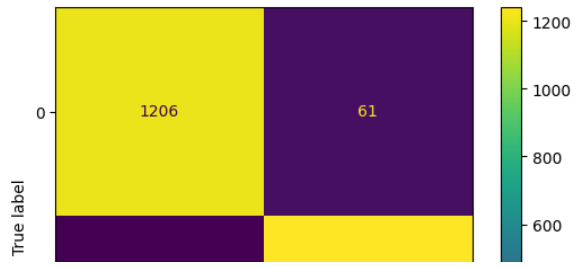
```
<6416x12589 sparse matrix of type '<class 'numpy.float64'>'
        with 19248 stored elements in Compressed Sparse Row format>
```

This code creates a display of what the mmodel

This code evaluates a KNN model's performance on a test dataset, printing the accuracy, displaying a confusion matrix, and printing a classification report. The unique labels for the classification report are extracted from the m2_pipeline's 'label' column.

```
#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

Double-click (or enter) to edit

The code extracts a subset of the original DataFrame mdf corresponding to rows labeled as "good trades" (where the 'label' column equals 1). The keepable list determines which columns should be retained in this subset, and the resulting DataFrame is stored in the variable ones_r.

```
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	prec
47	0.008169	-0.000036	0.003968	
108	-0.002202	-0.000064	-0.003449	
144	-0.204558	0.000178	-0.088451	
159	0.006487	-0.000134	0.003800	
189	0.001016	-0.000022	0.002570	
...	...	...	...	
6283	-0.012920	-0.003338	-0.002208	
6292	-0.008260	0.011126	-0.000741	
6312	0.215995	0.003481	0.027989	
6315	0.059232	0.009487	0.009968	
6395	0.005644	0.002875	0.000943	

119 rows × 7 columns

This code applies Granger Causality tests to a subset of data (ones_r) and generates a matrix (caul_mtrx) indicating whether one variable Granger-causes another. Only p-values less than or equal to 0.01 are shown in the final matrix.

```
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
```

Double-click (or enter) to edit

This code runs Granger Causality tests on a subset of data (zeroes_r), generates a matrix (caul_mtrx) indicating whether one variable Granger-causes another, and then filters the matrix to display only p-values less than or equal to 0.01. The distinction is that this is applied to a subset of data that corresponds to "bad trades" (where the 'label' column is 0).

```
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 unsu
  self._init_dates(dates, freq)
```

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bi
precursor_buy_cap_pct_change caused by	NaN	False	
precursor_ask_cap_pct_change caused by	False	NaN	
precursor_bid_vol_pct_change caused by	False	False	
precursor_ask_vol_pct_change caused by	False	False	
sum_change caused by	False	False	
length caused by	False	False	