premise: study underlying model dynamics

challenge assumptions on sell-factor causality in prices

install

This cell uses pip3 to install some Python libraries. These libraries help with functions such as data visualzation and statistical models.

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: portpicker in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair-vi
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
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->altai
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonsch
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->alta
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair-vi
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair->altair-viewe
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from portpicker->altair-data-server>=0.4.0
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.1
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.
Requirement already satisfied: portpicker in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair_
Requirement already satisfied: tornado in /usr/local/lib/python3.10/dist-packages (from altair-data-server>=0.4.0->altair_vi
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
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->altai
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in /usr/local/lib/python3.10/dist-packages (from jsonsch
Requirement already satisfied: referencing>=0.28.4 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair-
Requirement already satisfied: rpds-py>=0.7.1 in /usr/local/lib/python3.10/dist-packages (from jsonschema>=3.0->altair->alta
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair-
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.18->altair->altair_vi
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->altair->altair_viewe
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from portpicker->altair-data-server>=0.4.0
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas>=0.1
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (0.14.0)
Requirement already satisfied: numpy>=1.18 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.23.5)
Requirement already satisfied: scipy!=1.9.2,>=1.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.11.3)
Requirement already satisfied: pandas>=1.0 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.5.3)
Requirement already satisfied: patsy>=0.5.2 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.3)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (23.2)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0->statsmod
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0->statsmodels) (2023
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.2->statsmodels) (1.16.0)
Collecting imblearn
  Downloading imblearn-0.0-py2.py3-none-any.whl (1.9 kB)
Requirement already satisfied: imbalanced-learn in /usr/local/lib/python3.10/dist-packages (from imblearn) (0.10.1)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imblearn) (1
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imblearn) (1.
Requirement already satisfied: scikit-learn>=1.0.2 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imblea
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imblearn) (1
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from imbalanced-learn->imble
Installing collected packages: imblearn
Successfully installed imblearn-0.0
```

This code imports libraries needed for data manipulation, statistical analysis, and visualization.

```
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 cell loads a binary binned pipeline from a CSV file from the URL. Then it reads the CSV file into the pandas dataframe "mdf." It also shows the dataframe contents.

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

6416 rows × 39 columns

variables associated

This line of code retrieves the column labels and gives a list of the column names of the dataframe. This lets you work with specific columns.

```
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 cell filters the rows where the label column equals 1. It then calcuates the average of the surge targets met column for the filtered rows where the average is 0.74 or higher.

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

```
1.1650823181204584
```

This cell creates two dataframes where "ones" equals 1 and is marked as good trades and "zeros" has rows where the label equals 0 and is marked as bad trades.

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

study the underlying dichotomies in your data

This line of code retrieves the number of rows in the "ones" dataframe.

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

```
119
```

This line of code retrieves the number of rows in the "zeros" dataframe.

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

```
6297
```

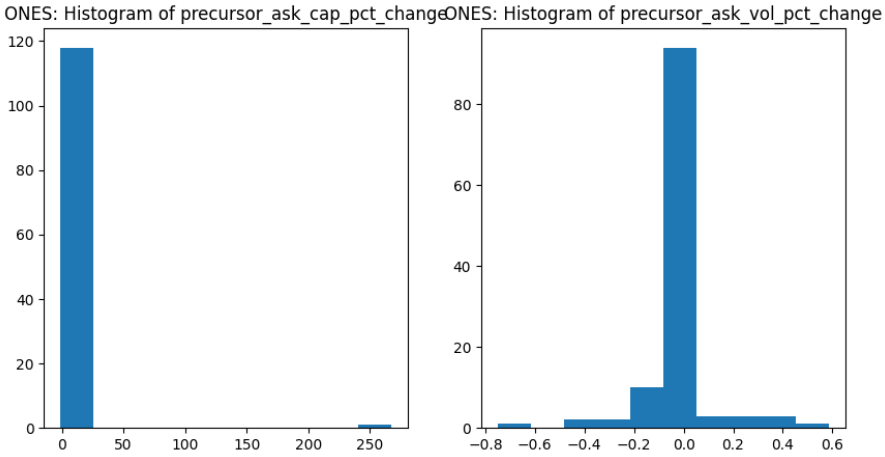
dimensions = features in the data we wish to study, before we classify

This cell defines a list named "dimensions." It has data of changes in ask prices and volumnes.

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

This cell creates two side by side plots with data from "ones" each having 10 bins. The first one is a histogram of "precursor_ask_cap_pct_change" data. The second shows the histogram for "precursor_ask_vol_pct_change."

```
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()
```

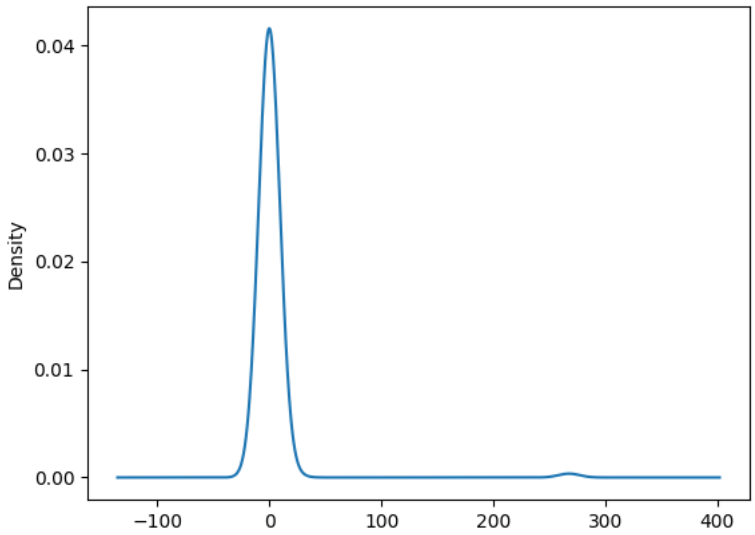


cap vs vol probability density, ONES

This code plots the density of the "precursor_ask_cap_pct_change" column from the "ones" dataframe.

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

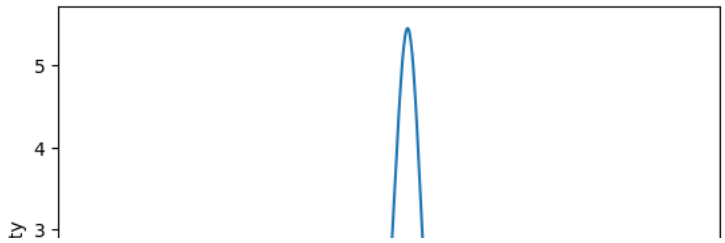
```
<Axes: ylabel='Density'>
```



This code plots the density of the "precursor_ask_vol_pct_change" column from the "ones" dataframe.

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

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

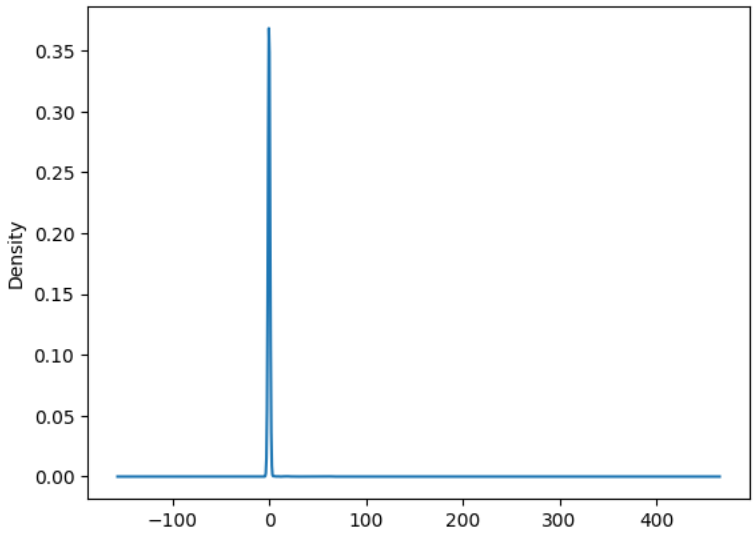


cap vs vol probability density, ZEROES

This code plots the density of the "precursor_ask_cap_pct_change" column from the "zeros" dataframe.

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

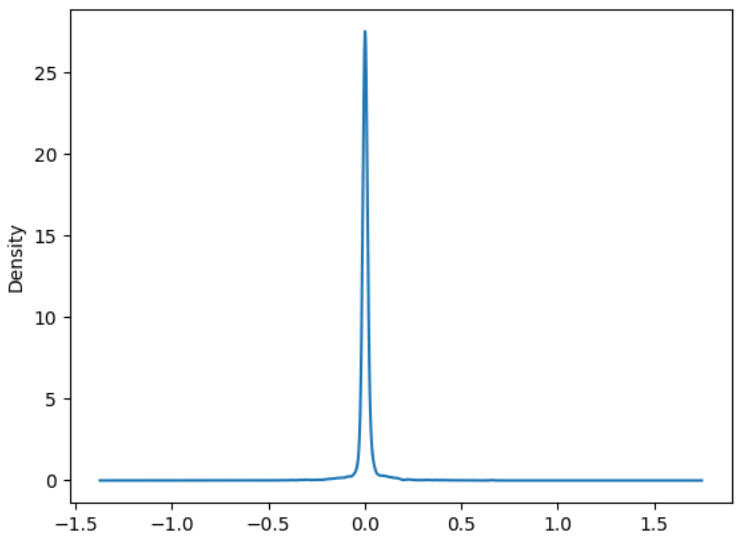
```
<Axes: ylabel='Density'>
```



This code plots the density of the "precursor_ask_vol_pct_change" column from the "ones" dataframe.

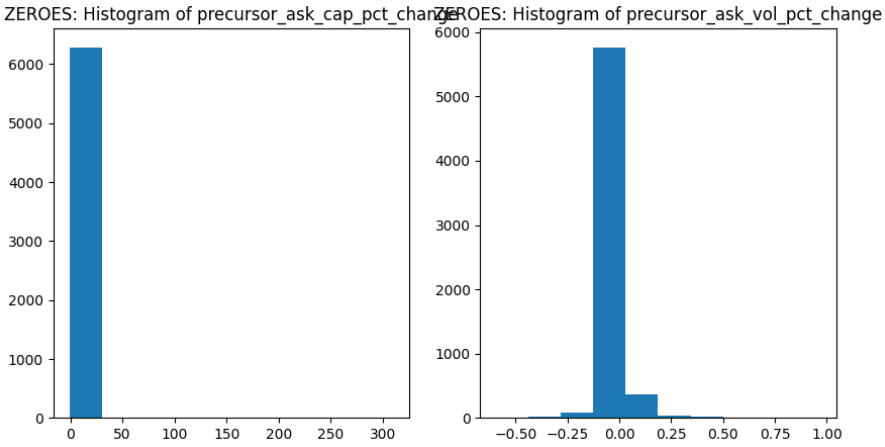
```
zeroes['precursor_ask_vol_pct_change'].plot.density()  #precursor_ask_vol_pct_change
```

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



This code creates two side by side histograms labeled "ZEROS" both with 10 bins and from the "zeros" dataset. One histogram uses data from the "precursor_ask_cap_pct_change" column and the other from the "precursor_ask_vol_pct_change" column.

```
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 chunk of code assigns a dataset "mdf" to the variable "m2_pipeline." Then a column name list is specified and the dataset is filtered to include the specified columns. A subset of columns is selected to create "X" and "y" where "X" has the selected feature values and "y" stands for labels.

```
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)
```

This cell selects columns in corr_list from m2_pipeline and creates a new dataframe. It then calculates the relation matrix for the selected columns in m2_pipeline using .corr().

```
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_chan
precursor_buy_cap_pct_change	1.000000	0.1959
precursor_ask_cap_pct_change	0.195900	1.0000
precursor_bid_vol_pct_change	0.547428	0.1909
precursor_ask_vol_pct_change	0.177817	0.2178
length	-0.074944	0.0552
sum_change	0.136782	-0.1316
surge_targets_met_pct	-0.001754	0.0679
time	-0.068998	-0.0447
label	-0.025531	0.0417

This cell imports pandas for data manipulation and matplotlib for plotting. It then makes a "df" dataaframe using m2_pipeline. It also makes a heatmap of the relation using the "matshow" function from matplotlib.

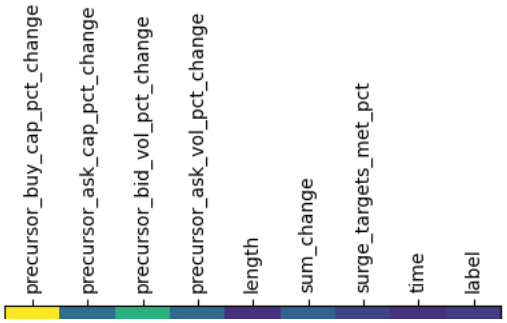
```
import pandas as pd
import matplotlib.pyplot as plt

# 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()
```



This cell makes predictions using a KNN model on the data stored in X_test_scaled and assigning it to 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()
```

This code computes the k-neighborhood graph for the data in X.

```
# 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 cell predicts labels for the data using KNN model and determines the accuracy of the predicitions. It also prints the accuracy score and gets certain labels used in the model. It also displays a confusion matrix and prints a classification report.

```
#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

This code selects rows where the label column equals 1 which is marked as good trades. The resulting data labeled as keepable 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_v
47	0.008169	-0.000036	
108	-0.002202	-0.000064	
144	-0.204558	0.000178	
159	0.006487	-0.000134	
189	0.001016	-0.000022	
...	...	...	
6283	-0.012920	-0.003338	
6292	-0.008260	0.011126	
6312	0.215995	0.003481	
6315	0.059232	0.009487	
6395	0.005644	0.002875	

119 rows × 7 columns

This code chunk makes a function named tes_granger which uses a VAR(p) model to make Granger Causality tests. Then it makes a new matrix where values less than or equal to 0.01 are marked as true and the rest marked as NaN.

```
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)
```

This cell filters the data for bad trades labeled as 0. It then tests the relationship among variables in the trades and renames the matrix by adding "caused by" to every index. It finally sets values to either NaN or <= 0.01 based on a conditional rule.

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

	precursor_buy_cap_pct_change	precursor_ask_cap_pct_change	precursor_bid_vol_pct_change	precursor
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	