1 HW: Machine Learning in Finance

1.1 due 2023-02-12

- Yu-Ching Liao ycliao3@illinois.edu (mailto:ycliao3@illinois.edu)

In [93]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

executed in 3ms, finished 14:29:52 2023-02-06

In [94]:
```
ds = pd.read_csv(
    "/Users/yu-chingliao/Library/CloudStorage/GoogleDrive-josephliao0127@gmail.com/My Drive/Note/UIUC/Spring_2023/IE51
)
ds
```

executed in 36ms, finished 14:29:53 2023-02-06

Out[94]:

	CUSIP	Ticker	Issue Date	Maturity	1st Call Date	Moodys	S_and_P	Fitch	Bloomberg Composite Rating	Coupon	...	percent_intra_dealer	percent_uncapped	bond
0	000324AA1	FLECIN	7/1/2014	7/1/2019	10/23/2017	Nan	Nan	Nan	Nan	12.00	...	0.006645	0.292359	
1	00080QAB1	RBS	3/15/2004	6/4/2018	Nan	Ba1	BB+	BBB	BB+	4.65	...	0.425018	0.974071	
2	00081TAD0	ACCO	5/14/2010	3/15/2015	Nan	WR	NR	BB+	NR	10.63	...	0.115207	0.594470	
3	00081TAH1	ACCO	6/17/2013	4/30/2020	Nan	WR	NR	WD	NR	6.75	...	0.426332	0.892462	
4	00081TAJ7	ACCO	12/22/2016	12/15/2024	12/15/2019	B1	BB-	BB	BB-	5.25	...	0.157216	0.690722	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2716	629377CC4	NRG	4/18/2017	1/15/2027	7/15/2021	B1	BB-	Nan	B+	6.63	...	0.376000	0.708571	
2717	62940QAA3	NSGHLD	3/14/2007	12/15/2025	Nan	Ba1	BB+	Nan	BB+	7.75	...	0.024540	0.699387	
2718	62941FAH1	VMED	7/25/2006	8/15/2016	Nan	WR	NR	BB+	NR	9.13	...	0.193798	0.527132	
2719	62943WAA7	NYLD	8/5/2014	8/15/2024	Nan	Ba2	BB	Nan	BB	5.38	...	0.063197	0.605948	
2720	62943WAB5	NYLD	7/21/2015	8/15/2024	8/15/2019	Ba2	BB	Nan	BB	5.38	...	0.241427	0.766118	

2721 rows × 37 columns

2 Print the shape out.

In [95]:
```
labels = list(ds.columns)
n_column = len(labels)
n_row = len(ds)

print("The number of Columns is", n_column, ".")
print("The number of Rows is", n_row, ".")
```

executed in 3ms, finished 14:29:55 2023-02-06

```
The number of Columns is 37 .
The number of Rows is 2721 .
```

3 Print the nature out

In [96]:
```
nl = []
sl = []
ol = []

for label in labels:
    Number = 0
    String = 0
    Other = 0

    for i in ds[label]:
        if type(i) == str:
            String += 1
        elif (type(i) == int) or (type(i) == float):
            Number += 1
        else:
            Other += 1
    nl.append(Number)
    sl.append(String)
    ol.append(Other)

Output = {
    "Label": labels,
    "Number": nl,
    "String": sl,
    "Other": ol
}
Output = pd.DataFrame(Output)
Output
```

executed in 52ms, finished 14:29:56 2023-02-06

Out[96]:

	Label	Number	String	Other
0	CUSIP	0	2721	0
1	Ticker	0	2721	0
2	Issue Date	0	2721	0
3	Maturity	0	2721	0
4	1st Call Date	0	2721	0
5	Moodys	0	2721	0
6	S_and_P	0	2721	0
7	Fitch	0	2721	0
8	Bloomberg Composite Rating	0	2721	0
9	Coupon	2721	0	0
10	Issued Amount	2721	0	0

4 Summary of Statistics

I pick column #9: Coupon as the example numerical data , and #12: Coupon Type as catagorical data.

In [118]:
```
numer = np.array(ds['Coupon'])

#Mean, Var and Std
print('μ =', numer.mean(), 'Var =', numer.var(), "σ =", numer.std(),'\n')

#quantiles
def q(ds, n_q):
    result = []
    for i in range(n_q+1):
        result.append(np.percentile(ds, i*(100)/n_q))
    return result
print("Boundaries for 4 Equal Percentiles\n",q(numer, 4), "\n")

#10 equal percenetiles
print("Boundaries for 10 Equal Percentiles\n",q(numer, 10), "\n")

#catagorical analysis
cat = list(ds['Coupon Type'])
neat_cat = list(set(cat))
print("Unique Label Values \n", neat_cat)

#count catagorics
counts = []
for i in neat_cat:
    counts.append(sum(ds['Coupon Type'] == i))
Output = {
    "Types" : neat_cat,
    "Counts" : counts
}
Output = pd.DataFrame(Output)
Output = Output.set_index("Types")
Output
```

executed in 21ms, finished 14:42:04 2023-02-06

```
μ = 10.30787210584344 Var = 3974.0157451596806 σ = 63.0397949327223

Boundaries for 4 Equal Percentiles
 [0.0, 5.0, 6.25, 7.75, 999.0]

Boundaries for 10 Equal Percentiles
 [0.0, 2.95, 4.63, 5.25, 5.75, 6.25, 6.83, 7.5, 8.13, 9.38, 999.0]

Unique Label Values
 ['FIXED', 'STEP CPN', 'DEFAULTED', 'EXCHANGED', 'VARIABLE', 'FUNGED', 'FLOATING', 'FLAT TRADING', 'PAY-IN-KIND', 'ZE
RO COUPON']
```

Out[118]:

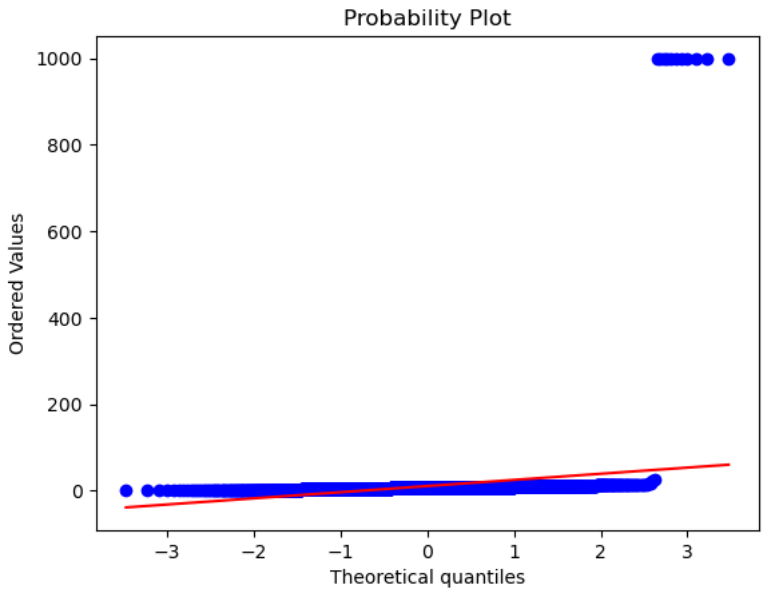
Types	Counts
FIXED	2139
STEP CPN	4
DEFAULTED	184
EXCHANGED	102
VARIABLE	111
FUNGED	2
FLOATING	124
FLAT TRADING	7
PAY-IN-KIND	41
ZERO COUPON	7

5 QQ Plot

In [119]:
```
import pylab
import scipy.stats as stats
```

executed in 677ms, finished 14:45:17 2023-02-06

In [121]:
```
stats.probplot(numer, dist="norm", plot=pylab)
pylab.show()
```
executed in 123ms, finished 14:46:14 2023-02-06



There are some extremely value though rest are quite "normal".

6 Print Summary of data

In [124]:
```
summary = ds.describe()
print(summary)
```
executed in 55ms, finished 14:52:25 2023-02-06

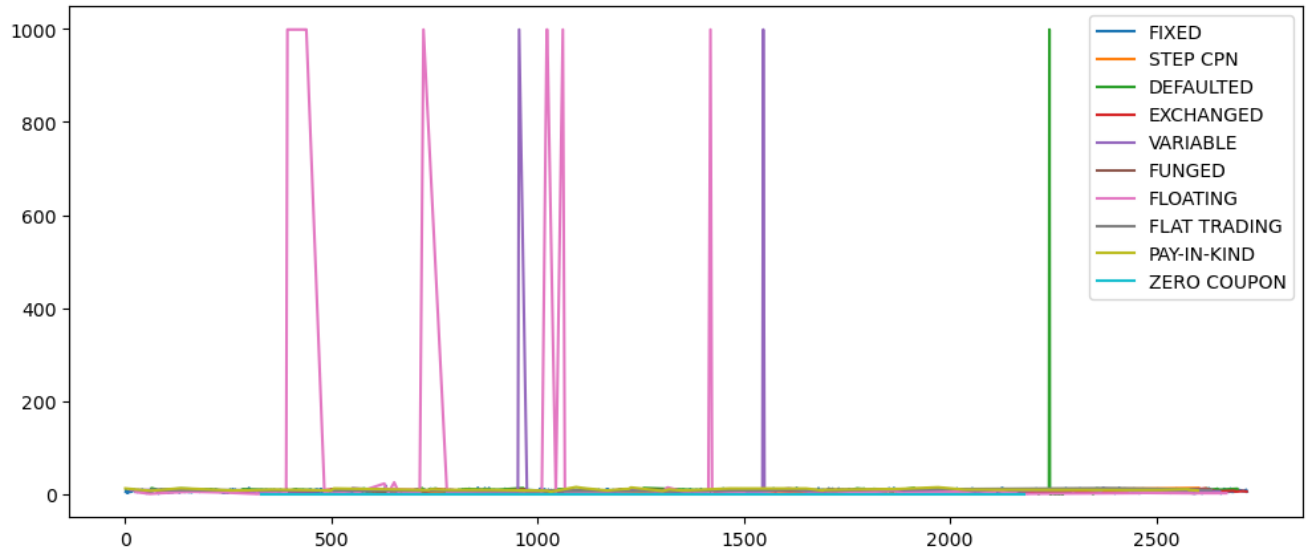
```
            Coupon  Issued Amount  Maturity At Issue months  LiquidityScore  \
count  2721.000000   2.721000e+03               2721.000000     2721.000000
mean     10.307872   8.299295e+08                113.968997       18.218230
std      63.051382   5.802790e+08                101.893176        7.872071
min       0.000000   3.700000e+08                 11.930000        4.388758
25%       5.000000   5.000000e+08                 65.170000       12.738630
50%       6.250000   6.500000e+08                 97.370000       16.538471
75%       7.750000   1.000000e+09                121.770000       22.120108
max     999.000000   7.364026e+09               1217.570000       54.673908

         LIQ SCORE      n_trades  volume_trades  total_median_size  \
count  2721.000000   2721.000000   2.721000e+03       2.721000e+03
mean      0.182182   2700.696435   7.222372e+08       5.361476e+05
std       0.078721   5572.262205   1.027825e+09       4.193546e+05
min       0.043888      1.000000   7.000000e+03       4.000000e+03
25%       0.127386    116.000000   6.189000e+07       7.500000e+04
50%       0.165385    674.000000   3.480000e+08       5.000000e+05
75%       0.221201   2467.000000   9.328420e+08       1.000000e+06
max       0.546739  57935.000000   8.979960e+09       3.400000e+06
```

7 Plot out data

In [147]:
```
def legend(pos="bottom",ncol=3):
    if pos=="bottom":
        plt.legend(bbox_to_anchor=(0.5,-0.2), loc='upper center',facecolor="lightgray",ncol=ncol)
    elif pos=="side":
        plt.legend(bbox_to_anchor=(1.1,0.5), loc='center left',facecolor="lightgray",ncol=1)
```
executed in 5ms, finished 15:06:13 2023-02-06

In [155]:
```
plt.figure(figsize=[12,5])
for i in neat_cat:
    plt.plot(ds['Coupon'].loc[ds['Coupon Type']==i], label = i)
plt.legend()
plt.show()
```

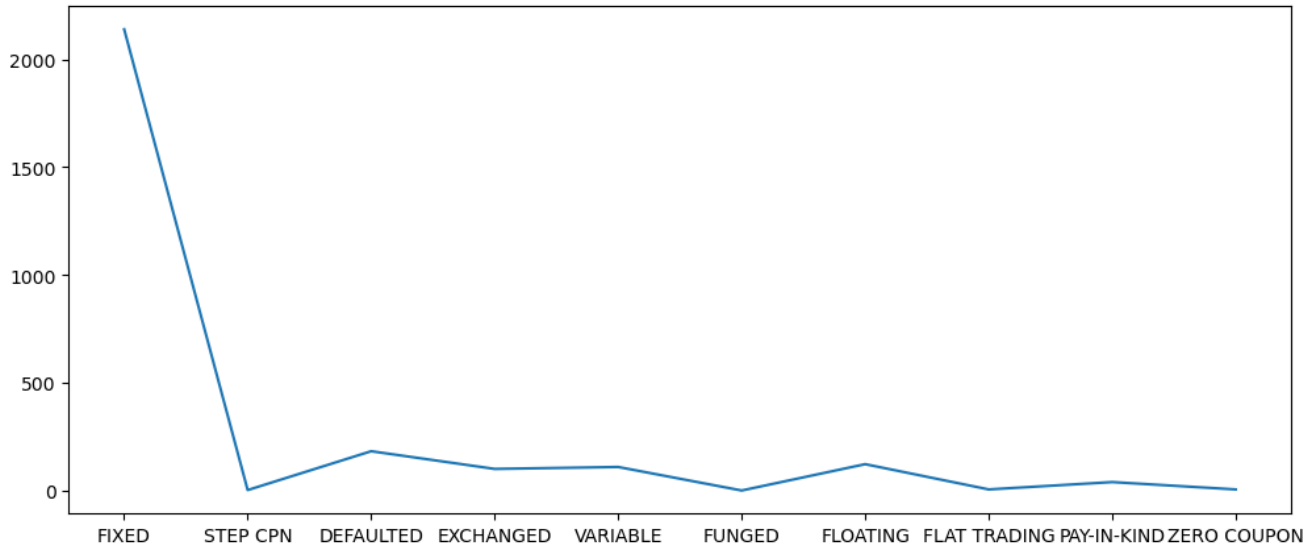
executed in 229ms, finished 15:15:33 2023-02-06



Since this plot is extremely useless, instead I plot below so that we can see the coupon we can get on different coupon types.

In [156]:
```
plt.figure(figsize=[12, 5])
plt.plot(Output)
plt.show()
```

executed in 148ms, finished 15:16:49 2023-02-06

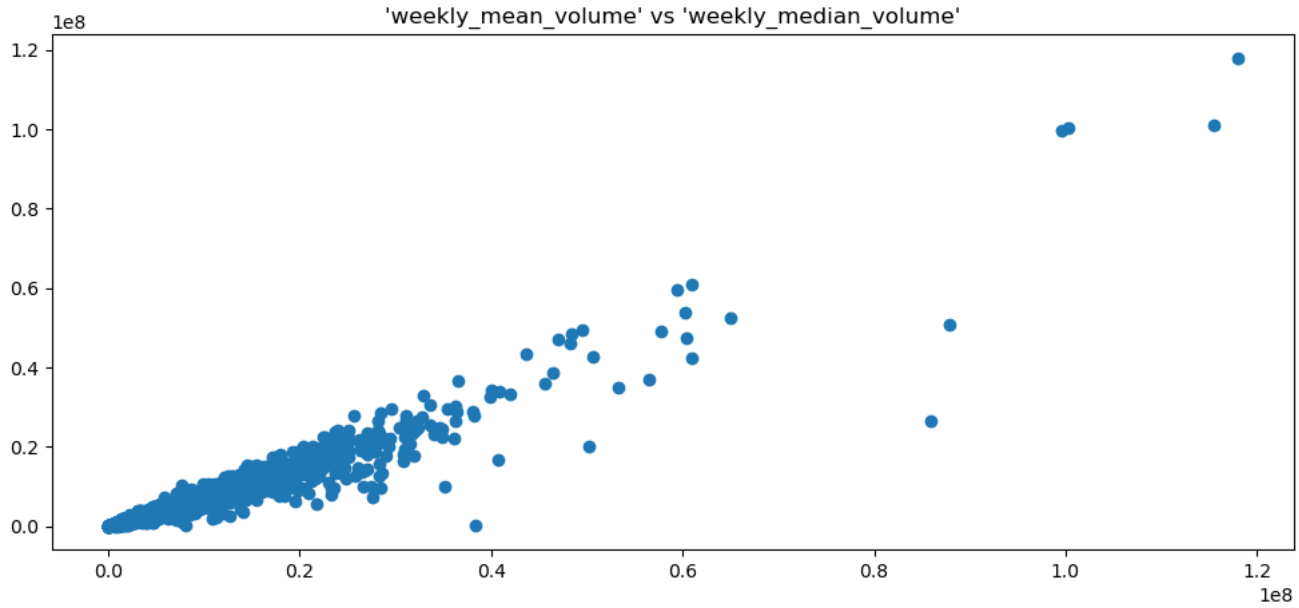


8 Cross Plotting Pairs of Attributes (Scatter Plot)

I use 'weekly_mean_volume' to cross plot with 'weekly_median_volume'.

In [170]:
```
plt.figure(figsize=[12,5])
plt.scatter(ds['weekly_mean_volume'], ds['weekly_median_volume'])
plt.title("'weekly_mean_volume' vs 'weekly_median_volume'")
plt.show()
```

executed in 171ms, finished 15:23:21 2023-02-06



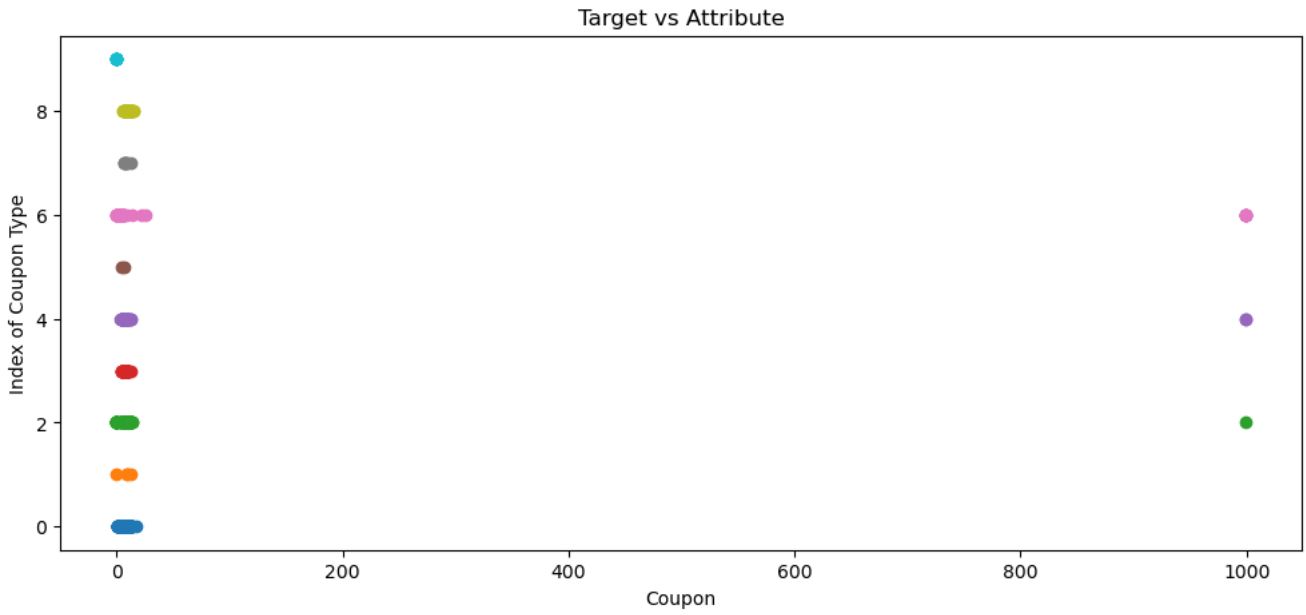
It is somewhat positively correlated, but not strict enough.

9 Target vs Real Attributes

I plot coupon against coupon types

In [181]:
```
plt.figure(figsize=[12,5])
for i in range(len(neat_cat)):
    plt.scatter(ds['Coupon'].loc[ds['Coupon Type'] == neat_cat[i]],
                i* np.ones_like(ds['Coupon'].loc[ds['Coupon Type'] == neat_cat[i]]))
plt.ylabel("Index of Coupon Type")
plt.xlabel("Coupon")
plt.title("Target vs Attribute")
plt.show()
```

executed in 254ms, finished 15:37:13 2023-02-06



10 Correlations

In [192]:
```
ds.corr()
```
executed in 38ms, finished 15:42:29 2023-02-06

Out[192]:

	Coupon	Issued Amount	Maturity At Issue months	LiquidityScore	LIQ SCORE	n_trades	volume_trades	total_median_size	total_mean_size	n_days_trade	.
Coupon	1.000000	-0.014238	0.098844	-0.042302	-0.042302	-0.023330	-0.026717	0.044601	0.026891	-0.028336	.
Issued Amount	-0.014238	1.000000	0.008601	0.134930	0.134930	0.156948	0.326310	0.062343	0.078362	0.068113	.
Maturity At Issue months	0.098844	0.008601	1.000000	0.072507	0.072507	0.038839	-0.015227	-0.115086	-0.138581	0.029530	.
LiquidityScore	-0.042302	0.134930	0.072507	1.000000	1.000000	0.803139	0.786718	-0.627008	-0.656980	0.873040	.
LIQ SCORE	-0.042302	0.134930	0.072507	1.000000	1.000000	0.803139	0.786718	-0.627008	-0.656980	0.873040	.
n_trades	-0.023330	0.156948	0.038839	0.803139	0.803139	1.000000	0.769322	-0.425801	-0.468673	0.704310	.
volume_trades	-0.026717	0.326310	-0.015227	0.786718	0.786718	0.769322	1.000000	-0.276204	-0.278564	0.772564	.
total_median_size	0.044601	0.062343	-0.115086	-0.627008	-0.627008	-0.425801	-0.276204	1.000000	0.930213	-0.490428	.
total_mean_size	0.026891	0.078362	-0.138581	-0.656980	-0.656980	-0.468673	-0.278564	0.930213	1.000000	-0.494483	.
n_days_trade	-0.028336	0.068113	0.029530	0.873040	0.873040	0.704310	0.772564	-0.490428	-0.494483	1.000000	.
days_diff_max	-0.025089	-0.008097	0.103178	0.717280	0.717280	0.497633	0.540932	-0.425033	-0.430947	0.796236	.
percent_intra_dealer	-0.014316	0.052617	0.104127	0.671903	0.671903	0.415695	0.387555	-0.650101	-0.679317	0.500944	.
percent_uncapped	-0.045897	-0.112369	0.100168	0.666321	0.666321	0.396880	0.241814	-0.826443	-0.862401	0.433119	.
bond_type	0.051856	-0.070714	0.102990	-0.368492	-0.368492	-0.208283	-0.452584	0.081332	0.086759	-0.444068	.
Client_Trade_Percentage	0.029125	-0.049513	-0.040186	-0.496127	-0.496127	-0.348408	-0.327922	0.486900	0.502385	-0.406258	.
weekly_mean_volume	-0.027724	0.382050	-0.023002	0.385978	0.385978	0.309053	0.503159	0.060608	0.052018	0.168114	.
weekly_median_volume	-0.028584	0.396947	-0.032868	0.371213	0.371213	0.285998	0.479018	0.053381	0.054723	0.169430	.
weekly_max_volume	-0.026362	0.261469	-0.017137	0.481142	0.481142	0.432955	0.616802	-0.066253	-0.087748	0.323094	.
weekly_min_volume	-0.014438	0.105208	-0.020392	0.025707	0.025707	-0.041335	-0.037198	0.138658	0.177104	-0.118874	.
weekly_mean_ntrades	-0.028045	0.274420	0.036729	0.673569	0.673569	0.804753	0.602055	-0.378970	-0.428501	0.416313	.
weekly_median_ntrades	-0.018326	0.188765	0.006573	0.239951	0.239951	0.234165	0.144272	-0.096283	-0.100327	0.026527	.

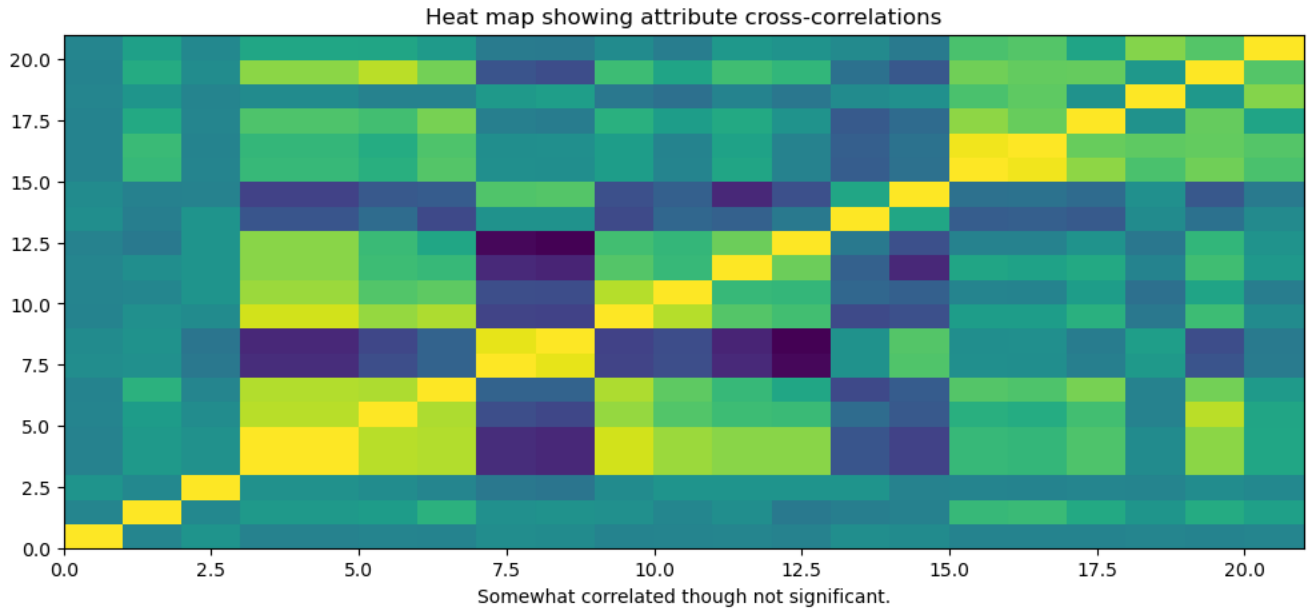
21 rows × 21 columns

n_trade and liquidity seemed to be positively correlated, and LIQ and liquidity is duplicating so redundant.

11 Correlation Visualization

In [188]:
```
#calculate correlations between real-valued attributes
corMat = pd.DataFrame(ds.corr())
#visualize correlations using heatmap
plt.figure(figsize=[12,5])
plt.title("Heat map showing attribute cross-correlations")
plt.pcolor(corMat)
plt.xlabel('Somewhat correlated though not significant.')
plt.show()
```

executed in 179ms, finished 15:40:00 2023-02-06



12 Signing

In [1]:
```
print("My name is Yu-Ching Liao")
print("My NetID is: 656724372")
print("I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.")
```

executed in 4ms, finished 15:57:22 2023-02-06

```
My name is Yu-Ching Liao
My NetID is: 656724372
I hereby certify that I have read the University policy on Academic Integrity and that I am not in violation.
```

In [ ]: