4D-Intelli

香 港 大 學
THE UNIVERSITY OF HONG KONG

FOMC Minutes Analysis

Can minutes forecast interest rate changes?

Group 4D-Intelli
Members:
Yuhang Xia, Chonwai Ho,
Hyun Jin Cho, Xueyang Wei,
Jiahao Sun, Cheng Peng,
Diyi Li
Mar 15th, 2019

CONTENT

Progress Check

Improvement in Data Processing

Data Analysis

Future Prospects

4D-Intelli

Improvement in Data Processing

Progress Check

Improvement in Data Processing

Data Analysis

Future Prospects

4D-Intelli

Improvement in Data Processing

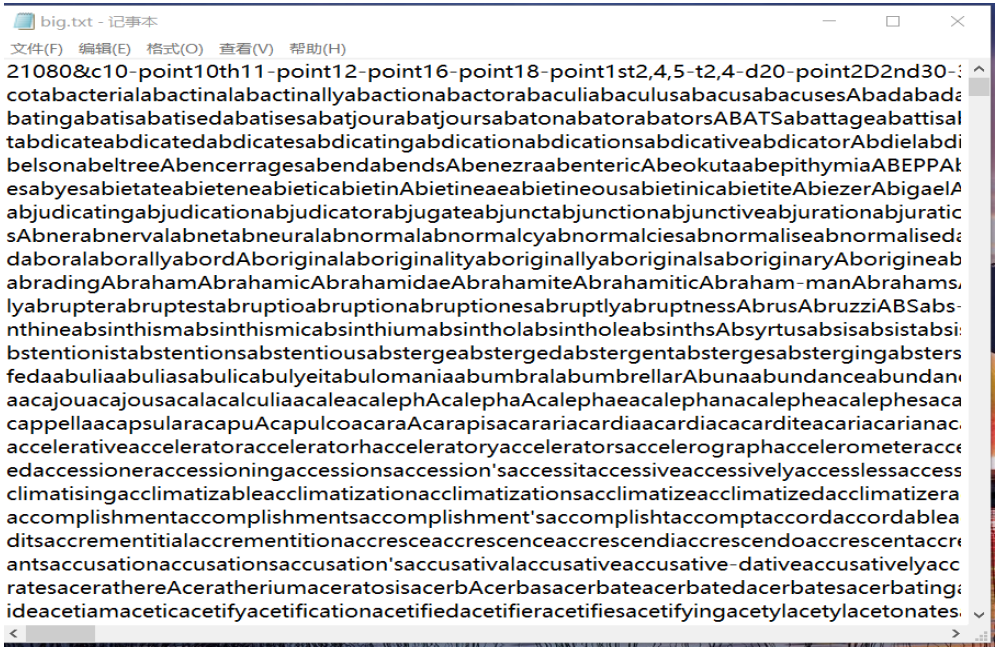
- Uninterrupted terms

Minutes of the Federal Open Market Committee June

June projectionsMarch projectionsNumber of participants

```
dictionary = Counter(words(open(r'C:\Users\sjh19\Desktop\big.txt').read()))
max_word_length = max(map(len, dictionary))
total = float(sum(dictionary.values()))
```

```
In [86]: len(dictionary)
Out[86]: 418914
```



4D-Intelli

Improvement in Data Processing

- Uninterrupted terms

```
dictionary = Counter(words(open(r'C:\Users\sjh19\Desktop\big.txt').read()))
max_word_length = max(map(len, dictionary))
total = float(sum(dictionary.values()))

def viterbi_segment(text):
    probs, lasts = [1.0], [0]
    for i in range(1, len(text) + 1):
        prob_k, k = max((probs[j] * word_prob(text[j:i]), j)
                        for j in range(max(0, i - max_word_length), i))
        probs.append(prob_k)
        lasts.append(k)
    words = []
    i = len(text)
    while 0 < i:
        words.append(text[lasts[i]:i])
        i = lasts[i]
    words.reverse()
    return words, probs[-1]
```

```
def lemmatization(data, allowed_postags=['NOUN', 'ADJ', 'VERB', 'ADV']):
    texts_out = []
    for sent in data:
        doc = nlp(re.sub('\_','',' ".join(sent)))
        tokens = [token.lemma_ for token in doc if token.pos_ in allowed_postags]
        new_tokens = []
        for i in tokens:
            for j in viterbi_segment(i)[0]:
                new_tokens.append(j)
        texts_out.append(new_tokens)
    return texts_out
```

4D-Intelli

Improvement in Data Processing

- Terms removing and standardizing

```
df_AnnualBow.astype(bool).sum(axis=1)
frequency = df_AnnualBow.astype(bool).sum(axis=0)
less = frequency[frequency<3]
df_AnnualBow=df_AnnualBow.drop(less.index,axis=1)
```

```
In [87]: frequency
Out[87]:
aa                6
aaa               5
aara              4
aaronson         11
aauw              1
ab                3
abandon           1
abandonment       1
abate            70
abatement        48
abel              2
```



```
In [98]: frequency[frequency>=3]
Out[98]:
aa                6
aaa               5
aara              4
aaronson         11
ab                3
abate            70
abatement        48
abet              9
ability         139
abl              19
able             65
```

4D-Intelli

Improvement in Data Processing

- Terms removing and standardizing

```
bowScaled = preprocessing.scale(df_AnnualBow)
df_bowScaled = pd.DataFrame(bowScaled,columns=df_AnnualBow.columns)
minutes_BoW_sk2 = pd.concat([minutes,df_bowScaled],axis = 1)
```

$Mean_i = 0$

$Std_i = 1$

```
In [100]: df_bowScaled
Out[100]:
          aa       aaa      aara  ...       zobe       zoe      zone
0  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
1  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
2  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
3  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
4  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
5  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
6  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
7  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
8  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
9  -0.112037 -0.097368 -0.091287  ...  -0.091287 -0.045502 -0.078975
```

Improvement in Data Processing

- Date mapping

Meeting date

2018 FOMC Meetings

January	30-31	**Statement:** PDF | HTML Implementation Note	Longer-Run Goals and Policy Strategy	**Minutes:** PDF | HTML (Released February 21, 2018)
March	20-21*	**Statement:** PDF | HTML Implementation Note	Press Conference **Projection Materials** PDF | HTML	**Minutes:** PDF | HTML (Released April 11, 2018)
May	1-2	**Statement:** PDF | HTML Implementation Note		**Minutes:** PDF | HTML (Released May 23, 2018)
June	12-13*	**Statement:** PDF | HTML Implementation Note	Press Conference **Projection Materials** PDF | HTML	**Minutes:** PDF | HTML (Released July 05, 2018)

Publication date

Improvement in Data Processing

- Date mapping

```
IR['Date'] = pd.to_datetime(IR['date'],format='%Y/%m/%d',errors='ignore')
minutes_BoW_sk2['Date'] = minutes_BoW_sk2['oldDate'] + datetime.timedelta(days=23)
minutes_tfidf_sk['Date'] = minutes_tfidf_sk['oldDate'] + datetime.timedelta(days=23)

bow_IR = pd.merge(IR,minutes_BoW_sk2,on = 'Date',how = 'left')
tfIdf_IR = pd.merge(IR,minutes_tfidf_sk,on = 'Date',how = 'left')
```

```
In [102]: bow_IR
Out[102]:
          date_x  fedRate       Date  ...       zoe
oldDate
0       1954/7/1     1.13 1954-07-01  ...       NaN
NaT
1       1954/7/2     1.25 1954-07-02  ...       NaN
NaT
2       1954/7/3     1.25 1954-07-03  ...       NaN
NaT
3       1954/7/4     1.25 1954-07-04  ...       NaN
NaT
4       1954/7/5     0.88 1954-07-05  ...       NaN
NaT
5       1954/7/6     0.25 1954-07-06  ...       NaN
```

```
...
23523  2019/2/20     2.40 2019-02-20  ...       NaN
NaT
23524  2019/2/21     2.40 2019-02-21  ...       NaN
NaT
23525  2019/2/22     2.40 2019-02-22  ... -0.045502
2019-01-30
23526  2019/2/25     2.40 2019-02-25  ...       NaN
NaT
23527  2019/2/26     2.40 2019-02-26  ...       NaN
```

23 days

4D-Intelli

Improvement in Data Processing

- Difference in Interest Rate

```
bow_IR_diff = bow_IR.dropna()
bow_IR_diff['rateChange'] = bow_IR_diff['fedRate'].shift(-1) - bow_IR_diff['fedRate']
```

```
In [105]: tfIdf_IR_diff['fedRate']
Out[105]:
4963    4.75
4991    4.75
5019    5.75
5028    5.75
5047    5.88
5064    6.25
5075    6.25
5103    6.38
5124    6.00
5152    6.13
5180    6.00
```

```
In [106]: tfIdf_IR_diff['fedRate'].shift(-1)
Out[106]:
4963    4.75
4991    5.75
5019    5.75
5028    5.88
5047    6.25
5064    6.25
5075    6.38
5103    6.00
5124    6.13
5152    6.00
5180    5.00
```

$$rateChange = IR_{i+1} - IR_i$$

4D-Intelli

Data Analysis

Progress Check

Improvement in Data Processing

Data Analysis

Future Prospects

4D-Intelli

single minutes interval, LSA for tfidf

Top words correlated with IR change

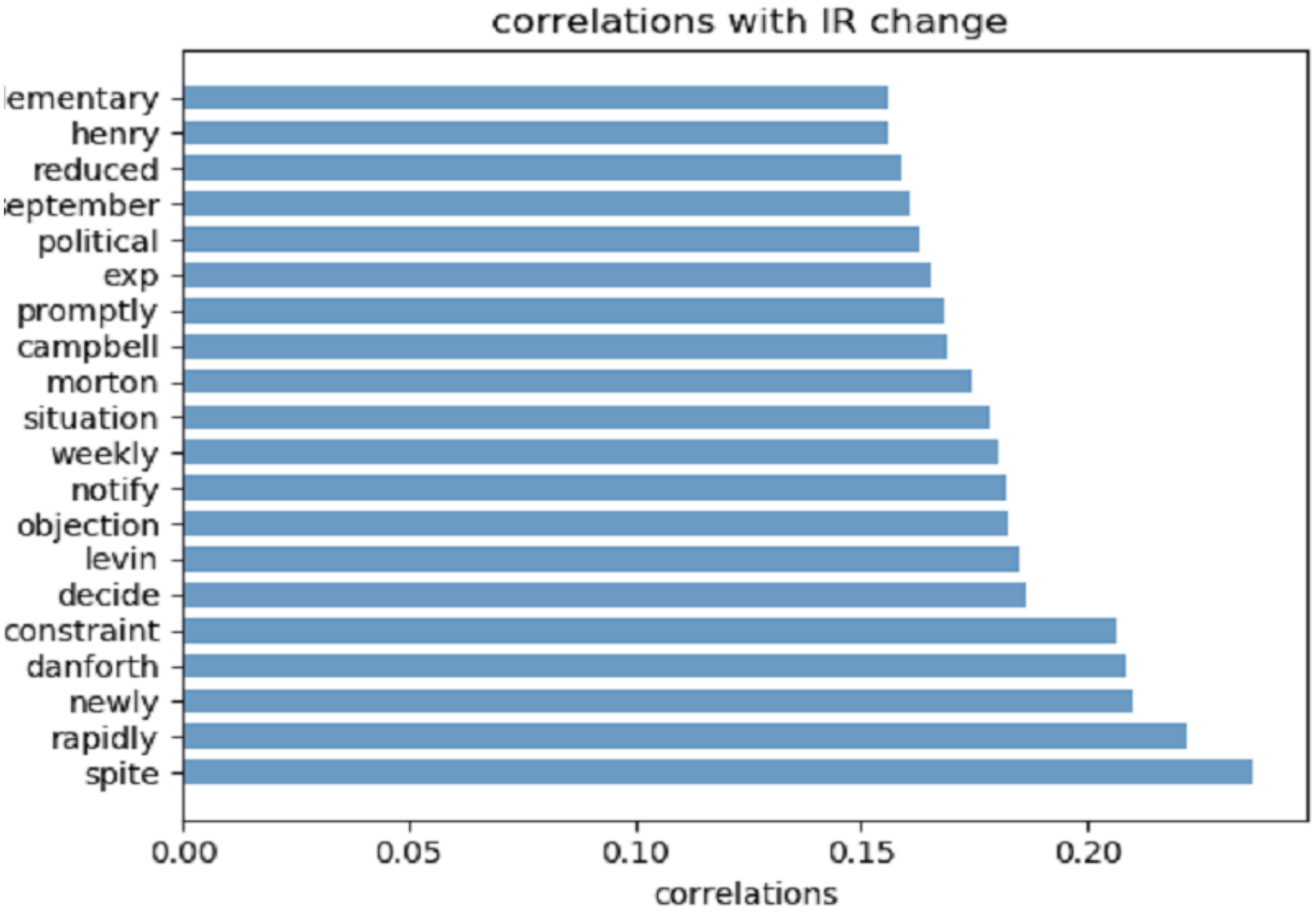
```
def CorTerms(terms,df_sum,y,top = None,bottom = None):
    correlations = [np.corrcoef(y,df_sum[term])[0,1]
                    for term in list(terms)]## change the index
    IR_corTerms = pd.DataFrame({'keyterms':terms,'correlations':correlations})
    top = IR_corTerms.sort_values(by = 'correlations',ascending = False) >> head(top)
    bottom = IR_corTerms.sort_values(by = 'correlations',ascending = True) >> head(bottom)
    return IR_corTerms,top,bottom

def corBar(x,y):
    plt.barh(range(len(x)), y, height=0.7, color='steelblue', alpha=0.8)
    plt.yticks(range(len(x)), x)
    plt.xlabel("correlations")
    plt.ylabel('keyterms')
    plt.title(" correlations with REIT change")
    plt.show()
```

4D-Intelli

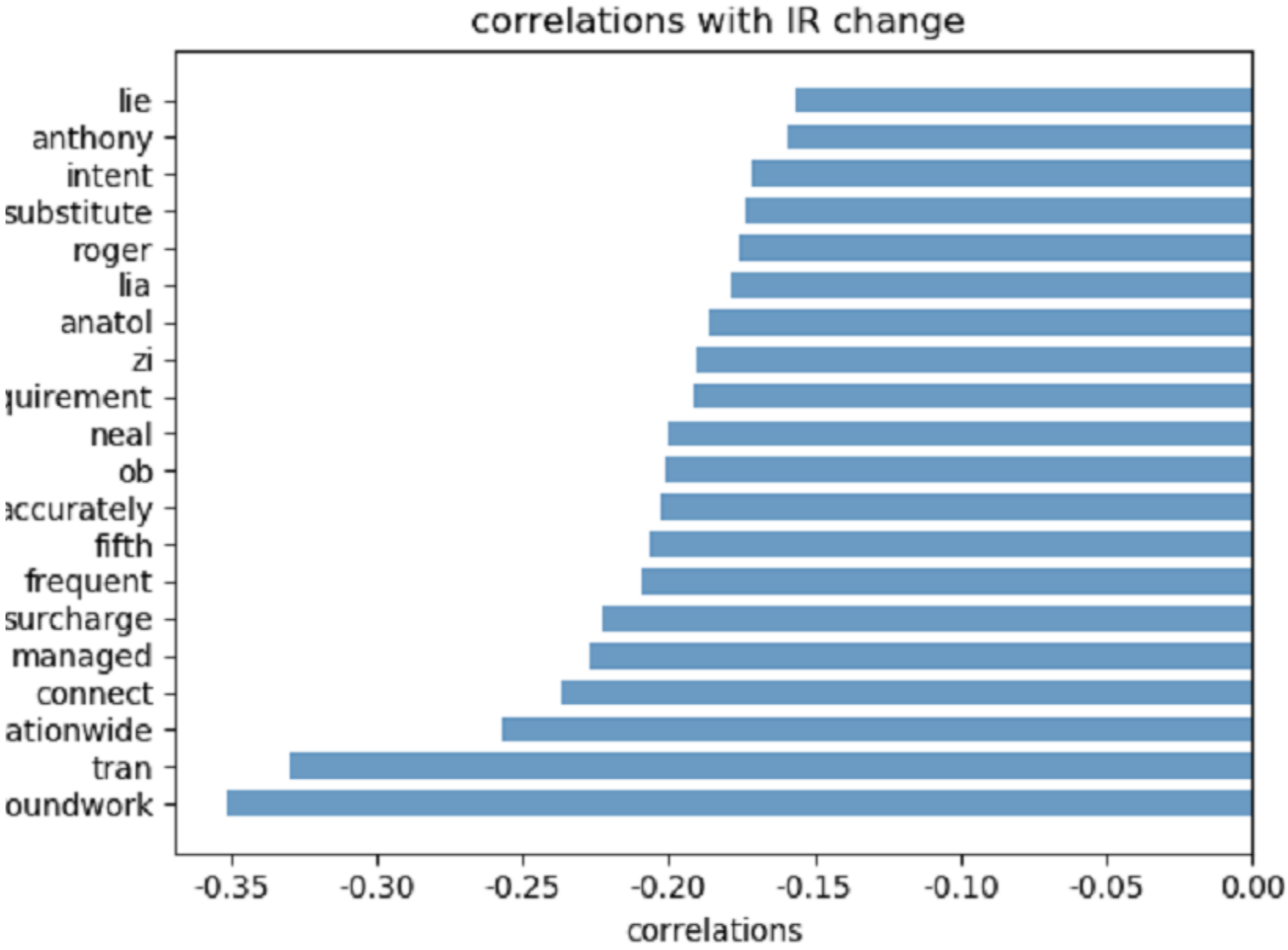
single minutes interval, LSA for tfidf

Top words correlated with IR change



single minutes interval, LSA for tfidf

bottom words correlated with IR change



single minutes interval, LSA for tfidf

Document similarity

```
#LSA

from sklearn.decomposition import TruncatedSVD
import sklearn
from sklearn.preprocessing import Normalizer
from sklearn import metrics
from sklearn.cluster import KMeans, MiniBatchKMeans
lsa=TruncatedSVD(400,algorithm='arpack')
df_tfidf_lsa = lsa.fit_transform(df_tfidfScaled)
```

single minutes interval, LSA for tfidf

Document similarity

```
tfIdf_IR_lsa = pd.merge(IR,minutes_tfidf_lsa,on = 'Date',how = 'left')
tfIdf_IR_diff_lsa = tfIdf_IR_lsa.dropna()
tfIdf_IR_diff_lsa['rateChange'] = tfIdf_IR_diff_lsa['fedRate'].shift(-1) - tfIdf_IR_diff_lsa['fedRate']
tfIdf_IR_diff_lsa = tfIdf_IR_diff_lsa.dropna()
tfIdf_IR_diff_lsa=tfIdf_IR_diff_lsa.drop(['year','month','day','file_name','oldDate','content'],axis=1)
CorTfidfIR_lsa = CorTerms(tfIdf_IR_diff_lsa.columns[3:-1],tfIdf_IR_diff_lsa,tfIdf_IR_diff_lsa['rateChange'])
tfIdf_top_lsa = CorTfidfIR_lsa[1]
tfIdf_bottom_lsa = CorTfidfIR_lsa[2]
corBar(tfIdf_top_lsa['keyterms'],tfIdf_top_lsa['correlations'])
corBar(tfIdf_bottom_lsa['keyterms'],tfIdf_bottom_lsa['correlations'])
import numpy as np
similarity = np.asarray(np.asmatrix(df_tfidf_lsa) * np.asmatrix(df_tfidf_lsa).T)
similarity = pd.DataFrame(similarity,index=minutes_tfidf_lsa['Date'], columns=minutes_tfidf_lsa['Date'])
```

single minutes interval, LSA for tfidf

Document similarity

命令提示符 - python

```
>>> tfIdf_bottom_lsa = CorTfidfIR_lsa[2]
>>> corBar(tfIdf_top_lsa['keyterms'],tfIdf_top_lsa['correlations'])
>>> corBar(tfIdf_bottom_lsa['keyterms'],tfIdf_bottom_lsa['correlations'])
>>> import numpy as np
>>> similarity = np.asarray(np.asmatrix(df_tfidf_lsa) * np.asmatrix(df_tfidf_lsa).T)
>>> similarity = pd.DataFrame(similarity,index=minutes_tfidf_lsa['Date'], columns=minutes_tfidf_lsa['Date'])
>>> similarity
```

Date	1968-02-01	1968-02-29	1968-03-28	1968-04-06	...	2018-10-19	2018-12-01	2019-01-11	2019-02-22
Date					...				
1968-02-01	1.000000	0.524546	0.052172	0.144785	...	-0.033528	-0.043631	-0.042663	-0.057050
1968-02-29	0.524546	1.000000	0.085647	0.122001	...	-0.047365	-0.049565	-0.055704	-0.062439
1968-03-28	0.052172	0.085647	1.000000	0.122759	...	-0.037024	-0.021472	-0.039610	-0.018640
1968-04-06	0.144785	0.122001	0.122759	1.000000	...	-0.014223	-0.038105	-0.021340	-0.037043
1968-04-25	0.272258	0.315226	0.138724	0.362354	...	-0.025794	-0.047433	-0.037163	-0.050250
1968-05-12	0.313416	0.489622	0.069366	0.312374	...	-0.041426	-0.067008	-0.049933	-0.084369
1968-05-23	0.363575	0.546925	0.127895	0.184213	...	-0.053365	-0.065654	-0.064549	-0.075595
1968-06-20	0.178772	0.219506	0.198839	0.116405	...	-0.038331	-0.045351	-0.043410	-0.042439
1968-07-11	0.205193	0.253697	0.044824	0.094730	...	-0.037249	-0.048814	-0.050742	-0.063034
1968-08-08	0.107384	0.152217	0.046524	0.078106	...	-0.031479	-0.020137	-0.023943	-0.021499
1968-09-05	0.227376	0.264092	0.030101	0.086418	...	-0.049944	-0.058171	-0.054948	-0.075031
1968-09-11	0.148380	0.214486	0.032197	0.191673	...	-0.036668	-0.063240	-0.044202	-0.076559
1968-10-03	0.251158	0.303069	0.037112	0.101574	...	-0.059089	-0.067570	-0.069718	-0.083600
1968-10-31	0.205664	0.305395	0.106878	0.336967	...	-0.052034	-0.066197	-0.058973	-0.076424
1968-11-21	0.248801	0.410942	0.071063	0.214735	...	-0.069968	-0.079773	-0.077982	-0.091683
1968-12-19	0.221312	0.316595	0.051272	0.128054	...	-0.056867	-0.064936	-0.065370	-0.076269
1969-01-09	0.256336	0.397828	0.048171	0.104327	...	-0.063348	-0.076641	-0.073625	-0.099555
1969-02-06	0.367817	0.306532	0.041735	0.102340	...	-0.055598	-0.075651	-0.065998	-0.095236
1969-02-27	0.234923	0.457365	0.032981	0.136309	...	-0.073199	-0.088027	-0.083632	-0.106065
1969-03-27	0.024207	0.015928	0.956425	0.063053	...	-0.041005	-0.032083	-0.042648	-0.021936
1969-04-24	0.230501	0.343687	0.039038	0.111283	...	-0.049485	-0.057919	-0.056287	-0.078503

4D-Intelli

single minutes interval, LSA for tfidf

Logistic regression to filter significant term

```
# logistic regression

IR_ChID= np.where(tfIdf_IR_diff_lsa['rateChange']>0,1,0)
tfIdf_IR_diff_lsa.insert(1,'IR_ChID',IR_ChID)
tfIdf_IR_diff_lsa
## to improve the model, I would filter out thoes insignifiant terms
from sklearn.feature_selection import f_regression
words = tfIdf_IR_diff_lsa.columns[4:-1]
X = tfIdf_IR_diff_lsa[words]
y = tfIdf_IR_diff_lsa['IR_ChID']
logisreg = f_regression(X, y, center=True)
Fvalue = logisreg[0]
Pvalue = logisreg[1]

stat_CorTfidfIR = CorTfidfIR_lsa[0]
stat_CorTfidfIR['Fvalue'] = Fvalue
stat_CorTfidfIR['Pvalue'] = Pvalue

signTerms = stat_CorTfidfIR.query('Pvalue < 0.05')
```

single minutes interval, LSA for tfidf

Logistic regression to filter significant term

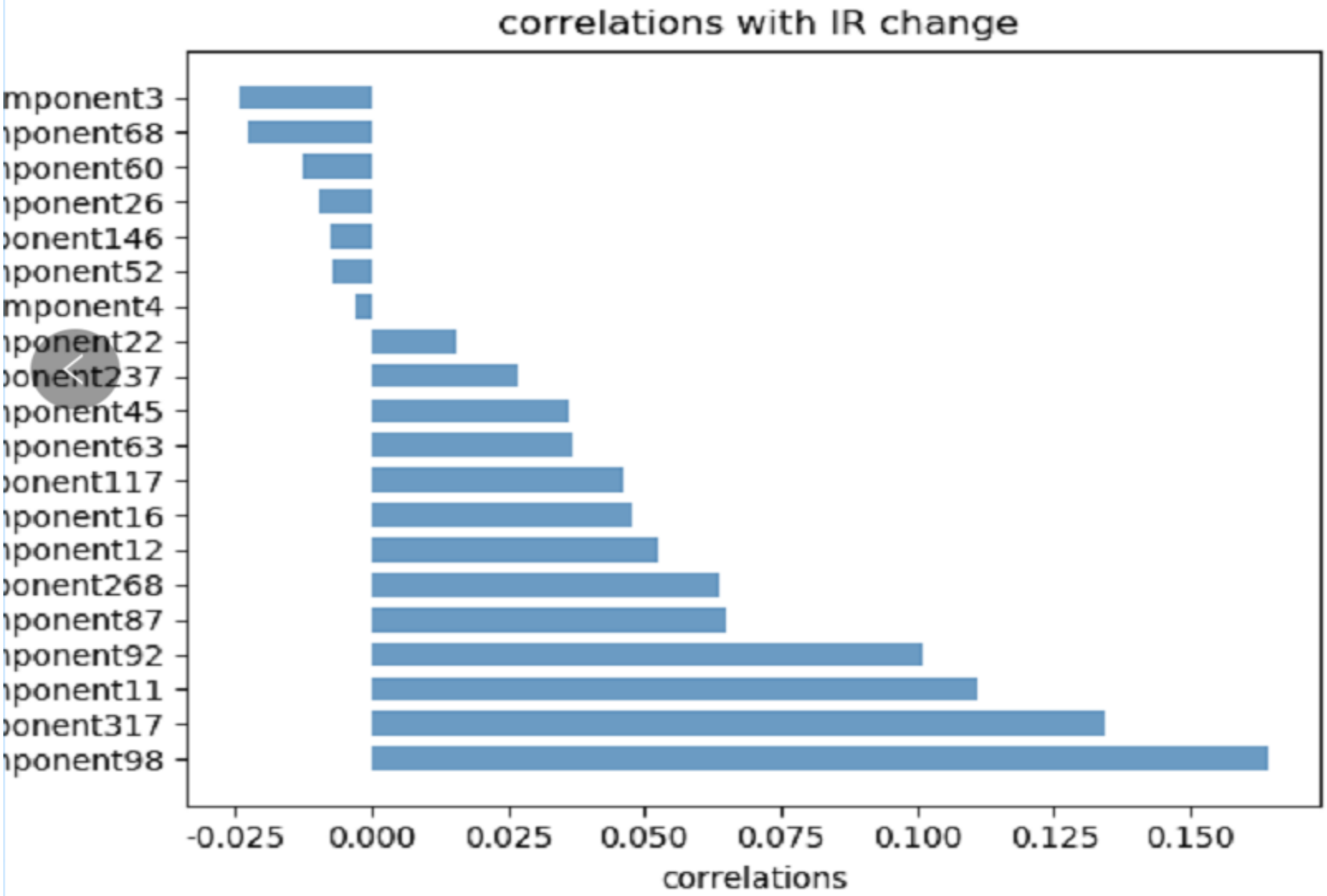
命令提示符 - python

```
>>> stat_CorTfidfIR['Fvalue'] = Fvalue
>>> stat_CorTfidfIR['Pvalue'] = Pvalue
>>>
>>>
>>> signTerms = stat_CorTfidfIR.query('Pvalue < 0.05')
>>>
>>> signTerms
        keyterms  correlations     Fvalue    Pvalue
2      component3     -0.024452   3.866337  0.049840
3      component4     -0.003047   4.605231  0.032375
10    component11      0.110961   4.325442  0.038077
11    component12      0.052415  10.257104  0.001452
14    component15     -0.078540   7.819303  0.005377
15    component16      0.047894   4.513499  0.034139
21    component22      0.015695   5.285012  0.021938
25    component26     -0.009605   8.142161  0.004512
37    component38     -0.044510   6.484734  0.011192
44    component45      0.036239   7.153953  0.007735
51    component52     -0.007287   3.986015  0.046444
59    component60     -0.012690   4.602177  0.032432
62    component63      0.036974   4.054925  0.044600
67    component68     -0.022549   5.847998  0.015966
86    component87      0.064773   6.944806  0.008678
91    component92      0.100788   6.548090  0.010805
97    component98      0.164345  11.128948  0.000916
116  component117      0.045986   8.689326  0.003357
144  component145     -0.083658   6.084578  0.013985
145  component146     -0.007527   6.328309  0.012208
191  component192     -0.065669   6.765959  0.009578
219  component220     -0.048392  11.138654  0.000911
```

4D-Intelli

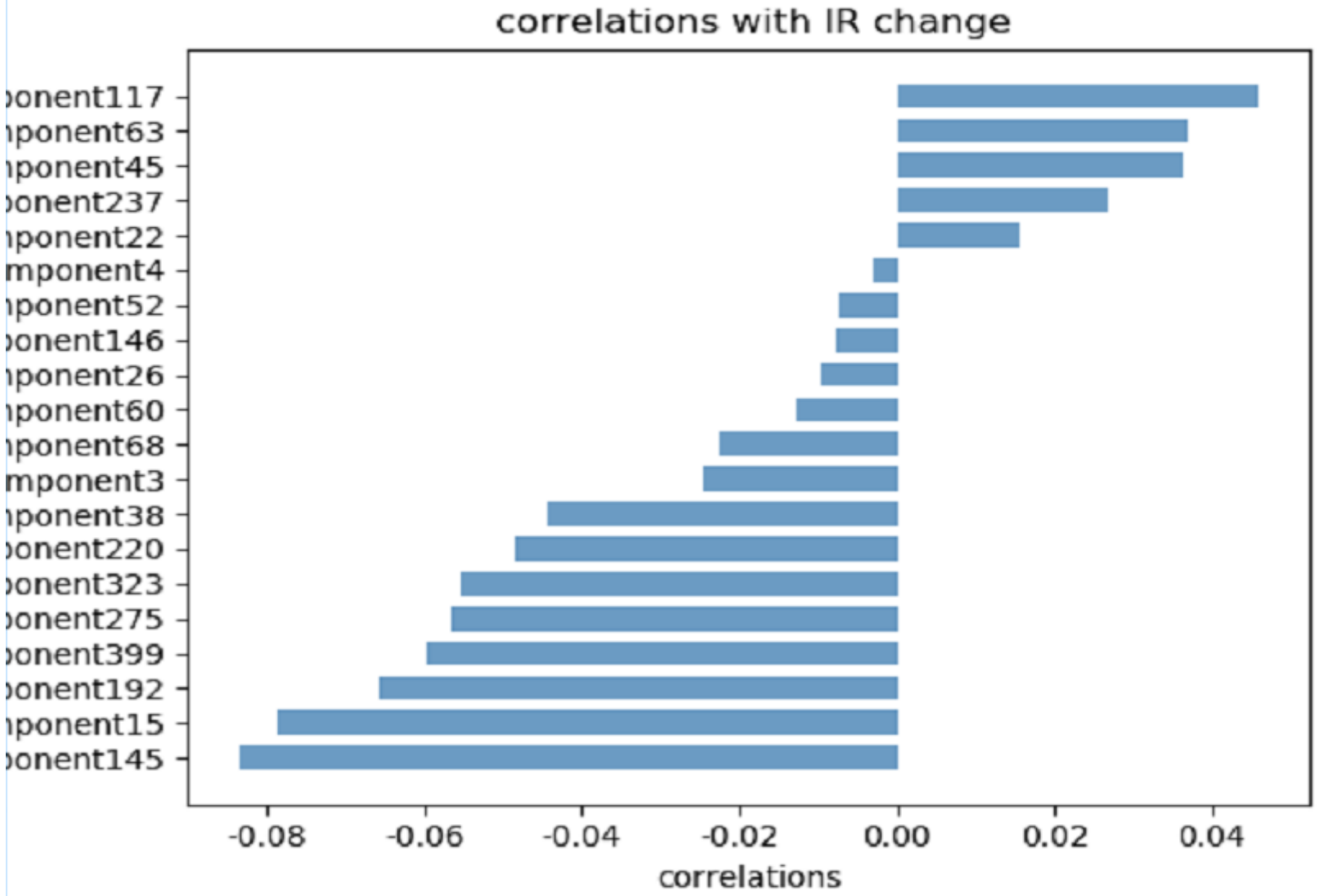
single minutes interval, LSA for tfidf

Logistic regression to filter significant term



single minutes interval, LSA for tfidf

Logistic regression to filter significant term



4D-Intelli

single minutes interval, LSA for tfidf

Machine learning to find the best prediction model

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
import time
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn import svm
from sklearn import metrics
import numpy as np
from sklearn import linear_model
from sklearn import svm
from sklearn import preprocessing
from sklearn import utils


ID_var = signTerms['keyterms'].tolist()
X = tfIdf_IR_diff_lsa[ID_var]
y= IR_ChID

lab_enc = preprocessing.LabelEncoder()
y_train_encoded = lab_enc.fit_transform(y_train)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

single minutes interval, LSA for tfidf

Machine learning to find the best prediction model

```
classifiers = {
    'SVR':svm.SVR(),
    'SVC':SVC(),
    'SGD':linear_model.SGDRegressor(),
    'BAYES':linear_model.BayesianRidge(),
    'LL':linear_model.LassoLars(),
    'ARD':linear_model.ARDRegression(),
    'PA':linear_model.PassiveAggressiveRegressor(),
    'TS':linear_model.TheilSenRegressor(),
    'L':linear_model.LinearRegression()
    }

train_scores = []
test_scores = []
names = []
models = {}
for key in classifiers.keys():
    clf = classifiers[key]
    clf.fit(X_train, y_train)
    train_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    y_test_predict = clf.predict(X_test)
    train_scores.append(train_score)
    test_scores.append(test_score)
    names.append(key)
```

single minutes interval, LSA for tfidf

Machine learning to find the best prediction model

	train_score	test_score	model
0	0.014356715	0.003141	SVR
1	0.503896104	0.56701	SVC
2	0.012373879	-0.00577	SGD
3	0.322599457	0.174375	BAYES
4	0	-0.01623	LL
5	0.320241901	0.083524	ARD
6	-3.241429167	-3.37246	PA
7	0.292898728	0.078041	TS
8	0.333467911	0.186075	L

single minutes interval, LSA for tfidf

Machine learning to find the best prediction model

```
for key in classifiers.keys():
  try:
    clf = classifiers[key]
    parameters = {'kernel':('linear', 'rbf'), 'C':(0.1, 1,5, 10)}
    gs = GridSearchCV(clf, parameters)
    gs.fit(X_train, y_train)
    best_scores.append(gs.best_score_)
    best_models.append(gs.best_params_)
    model_names.append(key)
  except:
    continue

best = {}
best['name'] = model_names
best['score'] = best_scores
best['param'] = best_models
df_best = pd.DataFrame(best)
df_best.to_csv('best.csv')

parameters = {'kernel':('linear', 'rbf'), 'C':(0.1,0.5,0.8, 1,1.2,1.5,2,5, 10)}
gs = GridSearchCV(SVC(), parameters)
gs.fit(X_train, y_train)
df_gs=pd.DataFrame(gs.cv_results_)
df_gs.to_csv('best2.csv')
gs.score(X_test,y_test)
```

single minutes interval, LSA for tfidf

Machine learning to find the best prediction model

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_tim	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.007537	0.000809	0.002704	0.002124	0.1	linear	{'C': 0.1, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
3	1	0.013337	0.004088	0.001065	0.000899	0.1	rbf	{'C': 0.1, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
4	2	0.005628	0.005244	0.004411	0.004896	0.5	linear	{'C': 0.5, 'k	0.713178	0.596899	0.614173	0.641558	0.051326	9	0.652344	0.652344	0.705426	0.6700
5	3	0.013599	0.004013	0.004998	0.00283	0.5	rbf	{'C': 0.5, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
6	4	0.007441	0.000107	0.002118	0.002995	0.8	linear	{'C': 0.8, 'k	0.697674	0.627907	0.645669	0.657143	0.029669	7	0.71875	0.710938	0.697674	0.7091
7	5	0.012751	0.001663	0.004932	0.001259	0.8	rbf	{'C': 0.8, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
8	6	0.007627	0.000719	0.00281	0.003267	1	linear	{'C': 1, 'kerı	0.682171	0.643411	0.661417	0.662338	0.015878	6	0.742188	0.722656	0.72093	0.7285
9	7	0.014876	3.32E-05	0.006199	0.00395	1	rbf	{'C': 1, 'kerı	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
10	8	0.008844	0.001051	0.002486	0.001847	1.2	linear	{'C': 1.2, 'k	0.697674	0.658915	0.685039	0.680519	0.016178	3	0.75	0.734375	0.713178	0.7325
11	9	0.010312	0.001657	0.005662	0.003333	1.2	rbf	{'C': 1.2, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
12	10	0.005602	0.003272	0.004271	0.00302	1.5	linear	{'C': 1.5, 'k	0.697674	0.666667	0.669291	0.677922	0.014062	4	0.753906	0.734375	0.717054	0.7351
13	11	0.010068	0.00227	0.00465	0.003615	1.5	rbf	{'C': 1.5, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	12	0.503906	0.503906	0.503876	0.5038
14	12	0.008804	0.002836	0.00135	0.000454	2	linear	{'C': 2, 'kerı	0.697674	0.674419	0.661417	0.677922	0.01499	4	0.761719	0.738281	0.732558	0.7441
15	13	0.010717	0.003357	0.003895	0.003284	2	rbf	{'C': 2, 'kerı	0.503876	0.503876	0.511811	0.506494	0.003731	11	0.503906	0.503906	0.53876	0.5155
16	14	0.006984	0.002583	0.005673	0.00123	5	linear	{'C': 5, 'kerı	0.713178	0.72093	0.677165	0.703896	0.019021	1	0.75	0.746094	0.736434	0.7441
17	15	0.013256	0.004343	0.007572	0.002751	5	rbf	{'C': 5, 'kerı	0.643411	0.573643	0.551181	0.58961	0.039274	10	0.582031	0.570313	0.624031	0.5921
18	16	0.008286	0.001651	0.003887	0.002796	10	linear	{'C': 10, 'ke	0.697674	0.72093	0.677165	0.698701	0.017858	2	0.773438	0.757813	0.748062	0.7597
19	17	0.011903	0.00251	0.003263	0.001505	10	rbf	{'C': 10, 'ke	0.689922	0.643411	0.637795	0.657143	0.023381	7	0.691406	0.703125	0.70155	0.6986
20																		
21																		
22																		

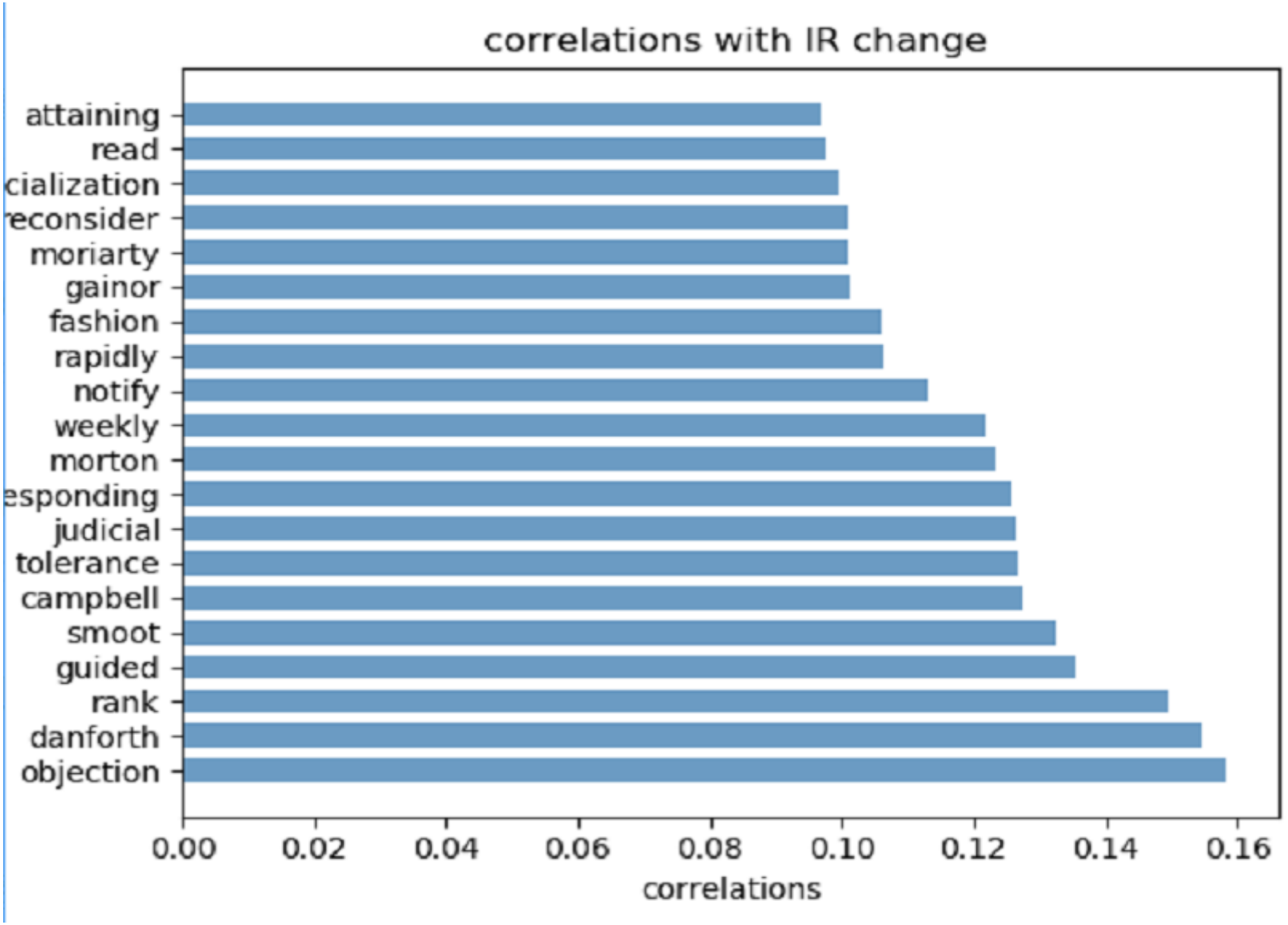
best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test,y_test)
0.6907216494845361
>>>
```

4D-Intelli

single minutes interval, LSA for bow

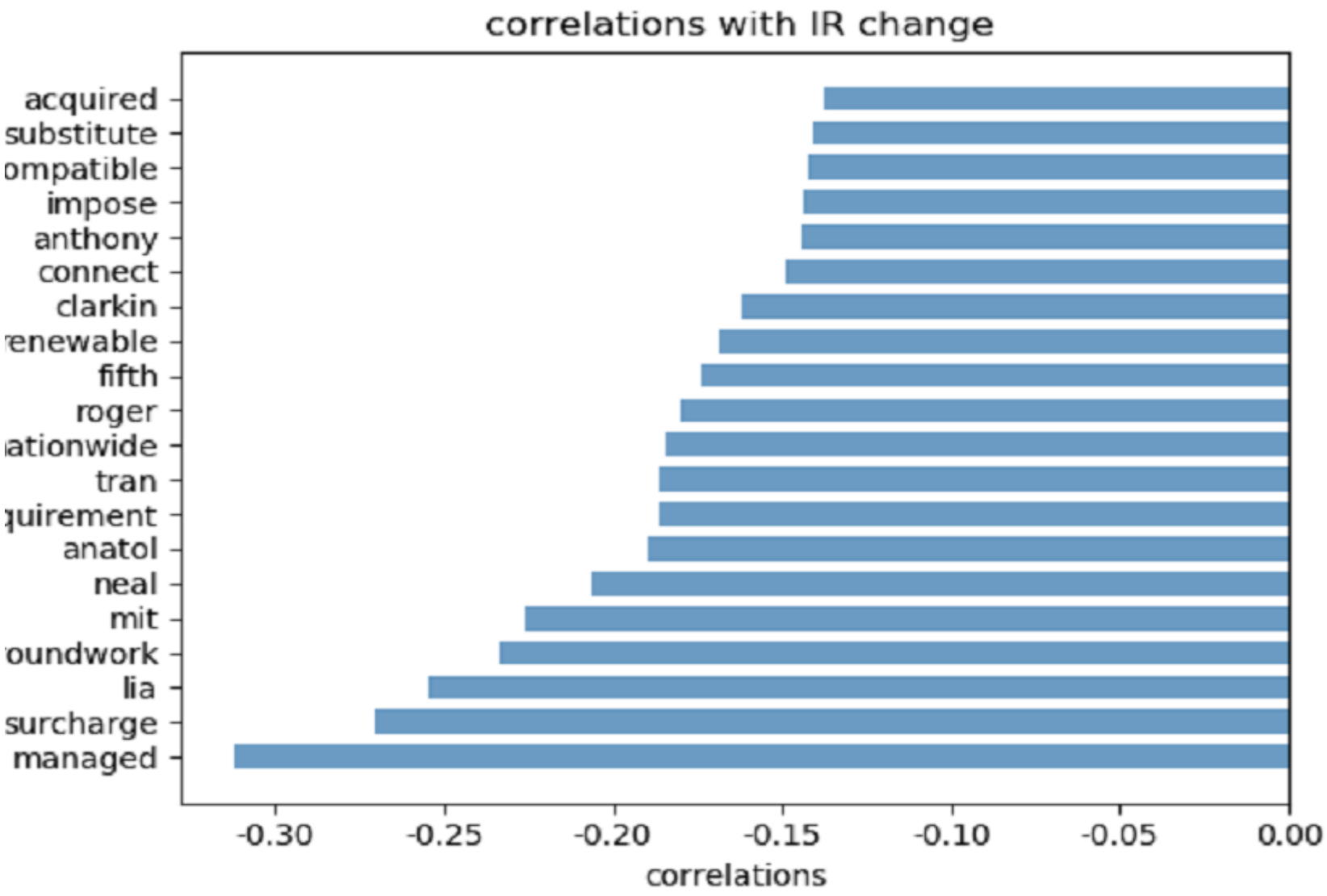
Correlation



4D-Intelli

single minutes interval, LSA for bow

Correlation



4D-Intelli

single minutes interval, LSA for bow

Similarity

命令提示符 - python

```
>>> bow_bottom_lsa = CorbowIR_lsa[2]
>>> corBar(bow_top_lsa['keyterms'],bow_top_lsa['correlations'])
>>> corBar(bow_bottom_lsa['keyterms'],bow_bottom_lsa['correlations'])
>>> import numpy as np
>>> similarity = np.asarray(np.asmatrix(df_bow_lsa) * np.asmatrix(df_bow_lsa).T)
>>> similarity = pd.DataFrame(similarity,index=minutes_bow_lsa['Date'], columns=minutes_bow_lsa['Date'])
>>>
>>> similarity
```

Date	1968-02-01	1968-02-29	1968-03-28	1968-04-06	...	2018-10-19	2018-12-01	2019-01-11	2019-02-22
Date					...				
1968-02-01	1.000000	0.698971	0.135387	0.303542	...	-0.131627	-0.098072	-0.151635	-0.160502
1968-02-29	0.698971	1.000000	0.202598	0.358710	...	-0.175254	-0.111389	-0.190547	-0.206823
1968-03-28	0.135387	0.202598	1.000000	0.194726	...	-0.038581	-0.016623	-0.038617	0.007451
1968-04-06	0.303542	0.358710	0.194726	1.000000	...	-0.111245	-0.089572	-0.118754	-0.116611
1968-04-25	0.509377	0.655214	0.258149	0.470619	...	-0.136277	-0.118351	-0.156499	-0.162678
1968-05-12	0.475147	0.662155	0.062130	0.547151	...	-0.208150	-0.162916	-0.231356	-0.254833
1968-05-23	0.584111	0.770181	0.235319	0.381901	...	-0.176555	-0.141839	-0.198771	-0.208729
1968-06-20	0.423974	0.560807	0.254641	0.337068	...	-0.154853	-0.122943	-0.167696	-0.166835
1968-07-11	0.443619	0.561995	0.139147	0.325912	...	-0.173142	-0.141680	-0.200359	-0.211731
1968-08-08	0.287205	0.401098	0.118295	0.209195	...	-0.109964	-0.058332	-0.108563	-0.110056
1968-09-05	0.442971	0.567198	0.093499	0.304692	...	-0.175586	-0.139131	-0.190652	-0.213992
1968-09-11	0.314720	0.463828	0.020130	0.393181	...	-0.182691	-0.142601	-0.201162	-0.222776
1968-10-03	0.497011	0.640422	0.093506	0.355484	...	-0.202883	-0.167575	-0.232645	-0.246981
1968-10-31	0.431125	0.624404	0.202010	0.491505	...	-0.180471	-0.132213	-0.194882	-0.210510
1968-11-21	0.489359	0.663825	0.163142	0.426193	...	-0.209929	-0.169651	-0.233742	-0.240929
1968-12-19	0.453122	0.608803	0.124086	0.351604	...	-0.186628	-0.153799	-0.212136	-0.222430
1969-01-09	0.419114	0.594332	0.107634	0.311040	...	-0.194020	-0.152974	-0.212177	-0.233135
1969-02-06	0.537673	0.548435	0.106911	0.309494	...	-0.190523	-0.154681	-0.209727	-0.231168
1969-02-27	0.428552	0.641792	0.074338	0.351949	...	-0.199084	-0.165909	-0.228284	-0.241716
1969-03-27	0.082894	0.096032	0.836640	0.125295	...	-0.046339	-0.034612	-0.045948	0.001384

4D-Intelli

single minutes interval, LSA for bow

Machine learning

	train_score	test_score	model
0	-0.107536343	-0.09046	SVR
1	0.503896104	0.56701	SVC
2	0.011256958	-0.00496	SGD
3	0.310577027	0.131098	BAYES
4	0	-0.01623	LL
5	0.314473966	0.021279	ARD
6	-0.058210353	-0.17806	PA
7	0.288455379	0.118623	TS
8	0.32838701	0.107693	L

single minutes interval, LSA for bow

Machine learning

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_tim	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.01006	0.002988	0.008406	0.003409	0.1	linear	{'C': 0.1, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
3	1	0.010473	0.002482	0.004908	0.004702	0.1	rbf	{'C': 0.1, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
4	2	0.00565	0.00397	0	0	0.5	linear	{'C': 0.5, 'k	0.604651	0.573643	0.535433	0.571429	0.028264	9	0.566406	0.582031	0.616279	0.5882
5	3	0.010919	0.003384	0.003761	0.005319	0.5	rbf	{'C': 0.5, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
6	4	0.007873	0.00483	0	0	0.8	linear	{'C': 0.8, 'k	0.651163	0.666667	0.582677	0.633766	0.036402	7	0.699219	0.636719	0.658915	0.6649
7	5	0.008295	0.002294	0.000678	0.000959	0.8	rbf	{'C': 0.8, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
8	6	0.000333	0.000471	0.001959	0.00277	1	linear	{'C': 1, 'ker	0.658915	0.682171	0.598425	0.646753	0.035218	6	0.6875	0.640625	0.689922	0.6726
9	7	0.01314	0.004672	0	0	1	rbf	{'C': 1, 'ker	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
10	8	0.005013	0.004174	0.004806	0.003829	1.2	linear	{'C': 1.2, 'k	0.689922	0.666667	0.590551	0.649351	0.042338	4	0.6875	0.675781	0.70155	0.6882
11	9	0.010225	0.004432	0.003753	0.00462	1.2	rbf	{'C': 1.2, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
12	10	0.000971	0.001373	0	0	1.5	linear	{'C': 1.5, 'k	0.674419	0.682171	0.590551	0.649351	0.041376	4	0.691406	0.707031	0.717054	0.7051
13	11	0.004657	0.00077	0	0	1.5	rbf	{'C': 1.5, 'k	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.503876	0.5038
14	12	0.005342	0.005729	0.004099	0.005108	2	linear	{'C': 2, 'ker	0.72093	0.682171	0.622047	0.675325	0.040607	3	0.695313	0.714844	0.724806	0.7116
15	13	0.006219	0.001852	0	0	2	rbf	{'C': 2, 'ker	0.503876	0.503876	0.503937	0.503896	2.87E-05	11	0.503906	0.503906	0.51938	0.5090
16	14	0.005025	0.00525	0.000333	0.000471	5	linear	{'C': 5, 'ker	0.697674	0.72093	0.645669	0.688312	0.031396	2	0.722656	0.722656	0.751938	0.7324
17	15	0.012423	0.0005	0.005104	0.001272	5	rbf	{'C': 5, 'ker	0.565891	0.542636	0.519685	0.542857	0.018839	10	0.546875	0.542969	0.581395	0.557
18	16	0	0	0.003347	0.004734	10	linear	{'C': 10, 'ke	0.705426	0.744186	0.629921	0.693506	0.047349	1	0.734375	0.746094	0.755814	0.7454
19	17	0.012564	0.003155	0.002895	0.004095	10	rbf	{'C': 10, 'ke	0.627907	0.573643	0.551181	0.584416	0.032203	8	0.609375	0.601563	0.651163	0.62
20																		
21																		
22																		

best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test,y_test)
0.6288659793814433
>>>
```

4D-Intelli

single minutes interval, signTerms for bow

Significance filtering

命令提示符 - python

```
>>>
>>> signTerms = stat_CorbowIR.query('Pvalue < 0.05')
>>> signTerms
          keyterms  correlations     Fvalue    Pvalue
14           abrupt      0.065347   9.307644  0.002408
15         abruptly     -0.004549   4.117933  0.042982
33       acceptance      0.009349   4.555901  0.033311
36      accommodate      0.042548   3.863591  0.049921
44       accordance      0.007089   4.067556  0.044271
46          account     -0.015274   5.974201  0.014876
49         accounts     -0.036721   3.965761  0.047001
56        accurately     -0.079223   3.989507  0.046348
81         addendum     -0.001088   4.968211  0.026280
91      adjournment     -0.017721   5.981885  0.014812
97   administration     -0.014437   5.713025  0.017224
104         advance      0.050983   5.073874  0.024740
105        advanced     -0.011256   4.805570  0.028847
107         adverse     -0.028003   7.749237  0.005586
109          advice     -0.016235   4.707763  0.030517
139       aggregate      0.015039   5.098103  0.024400
147       agreement     -0.006598   5.528492  0.019113
161            alan      0.015928   4.970718  0.026242
163             ale      0.029080   4.094866  0.043567
167          alfred      0.068873   5.486899  0.019567
186       alternate      0.002964   5.953964  0.015045
192        aluminum      0.012919   4.331298  0.037947
215        andersen      0.075585   6.471631  0.011273
252       anxieties      0.080906   6.089605  0.013946
258       apartment     -0.030350   5.022927  0.025470
275        applying     -0.014790   6.460742  0.011342
```

4D-Intelli

single minutes interval, signTerms for bow

Significance filtering

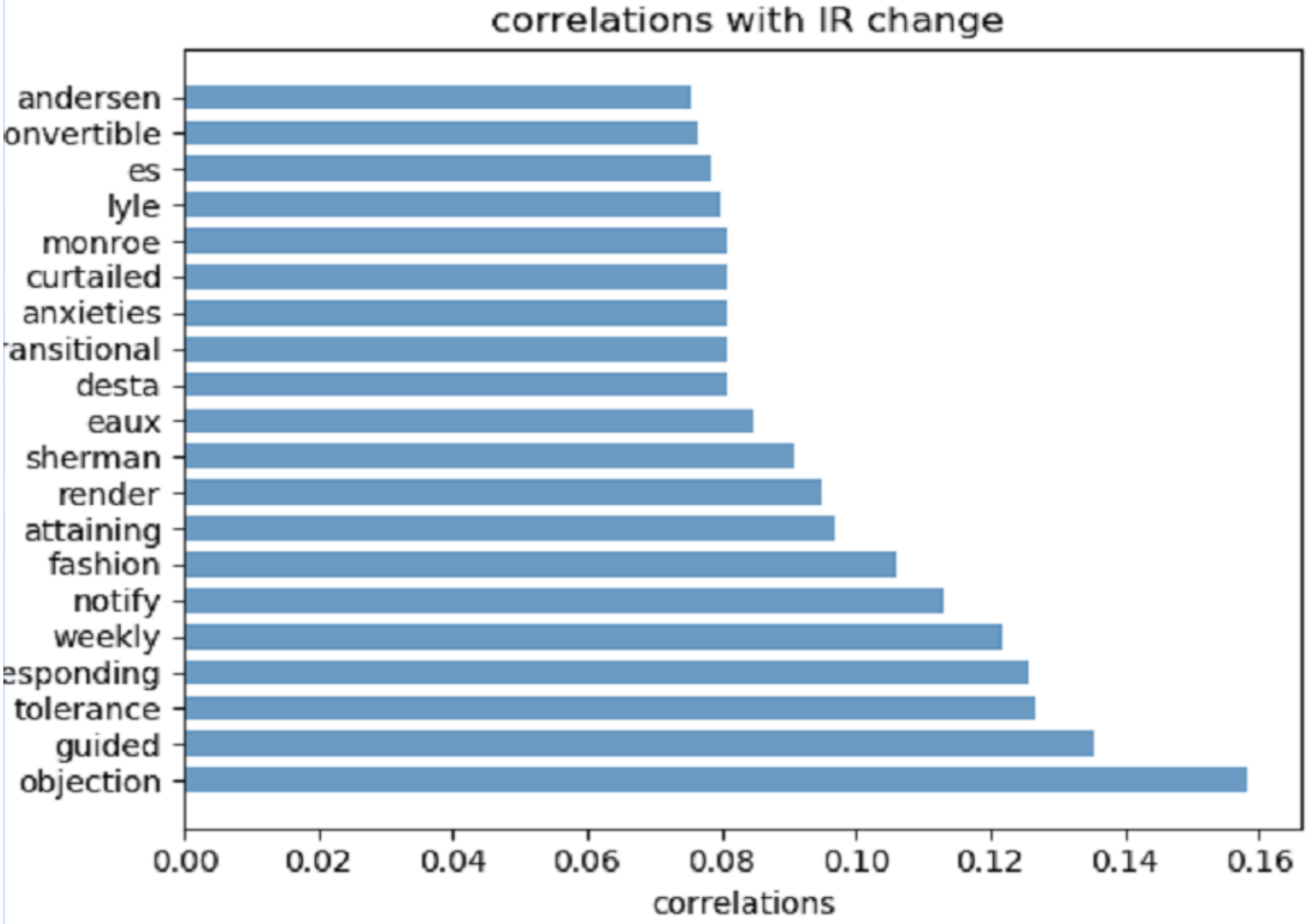
命令提示符 - python

5510	unless	0.014845	7.519540	0.006331
5511	unlikely	-0.004820	5.005247	0.025729
5523	unseasonably	0.014974	5.208892	0.022908
5535	unutilized	0.029788	4.331298	0.037947
5553	upturn	-0.029368	4.563560	0.033164
5567	usefully	-0.005211	7.702347	0.005730
5569	user	0.029000	5.987759	0.014763
5570	uso	0.000362	4.754779	0.029702
5586	valerie	0.001954	4.331298	0.037947
5638	vincent	0.025231	7.270650	0.007255
5655	vor	0.009264	4.331298	0.037947
5656	vote	0.014323	5.039313	0.025233
5661	vulnerability	0.035629	4.128265	0.042723
5664	wag	0.041394	5.437869	0.020117
5672	wallace	0.039567	8.892704	0.003009
5685	warm	0.006708	6.122167	0.013694
5687	warning	0.010431	5.437869	0.020117
5699	weak	-0.040371	4.859224	0.027972
5704	weakness	-0.077472	14.823714	0.000134
5712	weekly	0.121739	11.387097	0.000799
5716	weigh	-0.024027	5.636666	0.017981
5770	wonna	-0.107028	4.203591	0.040881
5787	worth	-0.005208	3.966428	0.046982
5788	worthy	-0.036014	5.902227	0.015488
5789	wright	0.002376	5.026852	0.025413
5799	yad	-0.014366	6.533316	0.010894
5809	york	-0.022408	4.502018	0.034367
5819	zina	-0.035955	4.203591	0.040881

[451 rows x 4 columns]

single minutes interval, signTerms for bow

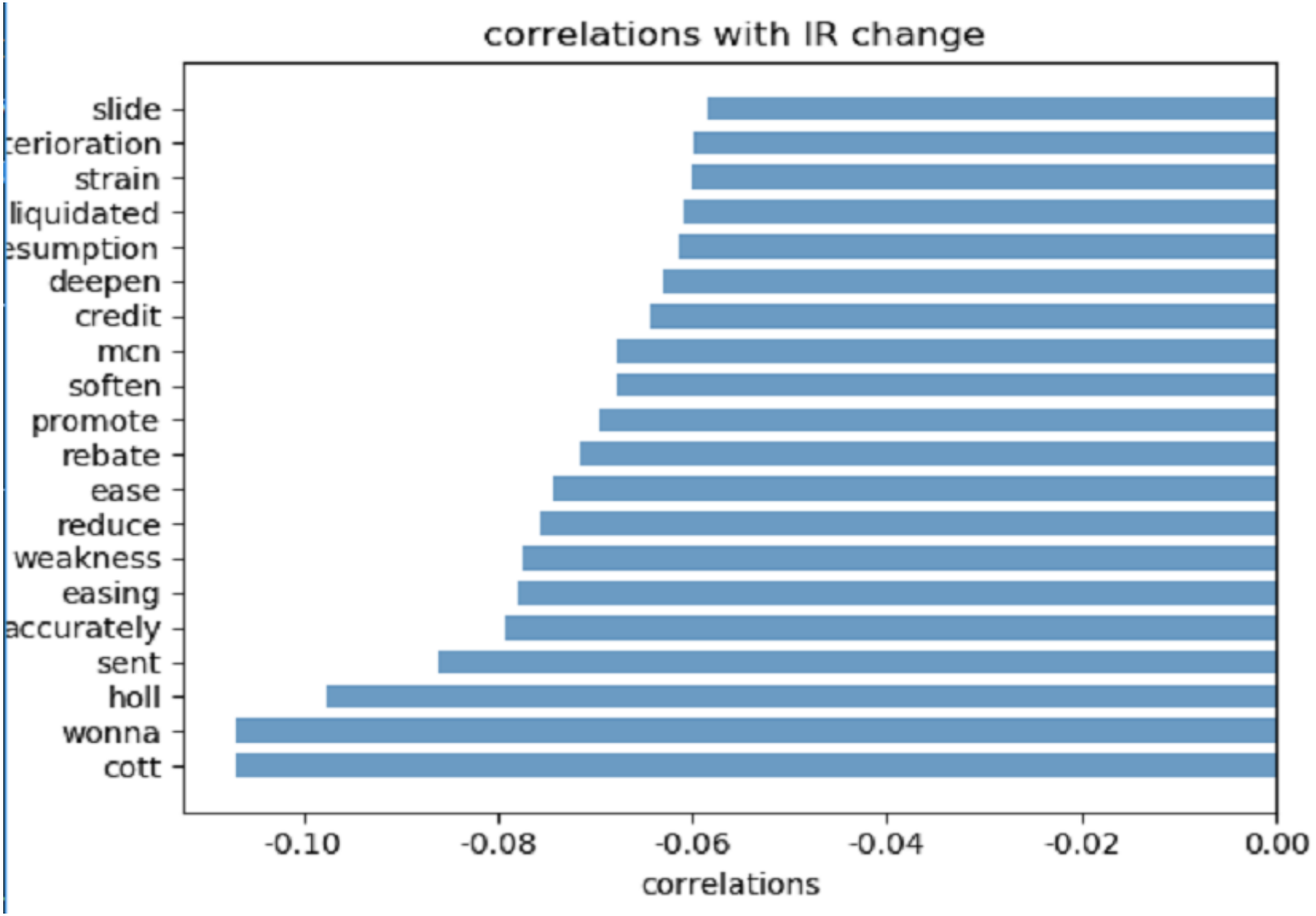
Correlation



4D-Intelli

single minutes interval, signTerms for bow

Correlation



4D-Intelli

single minutes interval, signTerms for bow

Model selection

	train_score	test_score	model
0	0.723671909	0.255501951	SVR
1	0.872727273	0.721649485	SVC
2	-5.10E+18	-1.54E+18	SGD
3	0.499311169	0.331920944	BAYES
4	0	-0.016225038	LL
5	0.613288499	-0.177685775	ARD
6	0.334369912	-0.609310768	PA
7	1	-733.9428491	TS
8	1	-734.3545906	L

single minutes interval, signTerms for bow

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_time	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.058913	0.005676	0.025636	0.002504	0.1	linear	{'C': 0.1, 'k	0.705426	0.736434	0.669291	0.703896	0.027396	4	0.945313	0.964844	0.968992	0.9597
3	1	0.085136	0.010328	0.045605	0.001956	0.1	rbf	{'C': 0.1, 'k	0.503876	0.503876	0.551181	0.519481	0.022241	18	0.503906	0.503906	0.627907	0.545
4	2	0.064079	0.018099	0.022824	0.003273	0.5	linear	{'C': 0.5, 'k	0.666667	0.666667	0.661417	0.664935	0.002468	12	0.992188	0.988281	0.996124	0.9921
5	3	0.091995	0.002655	0.043526	0.004264	0.5	rbf	{'C': 0.5, 'k	0.689922	0.674419	0.590551	0.651948	0.043541	16	0.753906	0.796875	0.77907	0.7766
6	4	0.057987	0.006292	0.020491	0.003051	0.8	linear	{'C': 0.8, 'k	0.666667	0.627907	0.661417	0.651948	0.0172	16	1	0.992188	0.996124	0.9961
7	5	0.089861	0.002133	0.04274	0.001681	0.8	rbf	{'C': 0.8, 'k	0.728682	0.689922	0.614173	0.677922	0.047457	7	0.839844	0.851563	0.860465	0.8506
8	6	0.071337	0.023152	0.021677	0.003497	1	linear	{'C': 1, 'kern	0.689922	0.666667	0.653543	0.67013	0.015034	9	1	0.996094	0.996124	0.9974
9	7	0.098913	0.009389	0.046591	0.009012	1	rbf	{'C': 1, 'kern	0.751938	0.697674	0.653543	0.701299	0.040198	5	0.871094	0.871094	0.875969	0.8727
10	8	0.068091	0.020335	0.024233	0.003033	1.2	linear	{'C': 1.2, 'k	0.689922	0.682171	0.637795	0.67013	0.022907	9	1	0.996094	0.996124	0.9974
11	9	0.093657	0.003046	0.043222	0.001712	1.2	rbf	{'C': 1.2, 'k	0.75969	0.713178	0.677165	0.716883	0.033748	3	0.878906	0.886719	0.887597	0.8844
12	10	0.068287	0.019625	0.022425	0.000489	1.5	linear	{'C': 1.5, 'k	0.689922	0.689922	0.637795	0.672727	0.024508	8	1	1	0.996124	0.9987
13	11	0.092835	0.002609	0.042078	0.003094	1.5	rbf	{'C': 1.5, 'k	0.782946	0.728682	0.661417	0.724675	0.049629	2	0.890625	0.890625	0.899225	0.8934
14	12	0.068979	0.018976	0.019628	0.003494	2	linear	{'C': 2, 'kern	0.689922	0.666667	0.637795	0.664935	0.021288	12	1	1	1	
15	13	0.09083	0.002799	0.042531	0.001572	2	rbf	{'C': 2, 'kern	0.79845	0.744186	0.637795	0.727273	0.066591	1	0.917969	0.898438	0.914729	0.9103
16	14	0.055425	0.008028	0.01912	0.002589	5	linear	{'C': 5, 'kern	0.689922	0.666667	0.622047	0.65974	0.028106	14	1	1	1	
17	15	0.091953	0.002696	0.039433	0.003834	5	rbf	{'C': 5, 'kern	0.728682	0.697674	0.622047	0.683117	0.044687	6	0.933594	0.921875	0.934109	0.9298
18	16	0.067019	0.020954	0.021563	0.003657	10	linear	{'C': 10, 'ke	0.689922	0.666667	0.622047	0.65974	0.028106	14	1	1	1	
19	17	0.079092	0.012109	0.037639	0.009472	10	rbf	{'C': 10, 'ke	0.713178	0.674419	0.622047	0.67013	0.037279	9	0.945313	0.957031	0.972868	0.9584
20																		
21																		
22																		

best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test,y_test)
0.7525773195876289
>>>
```

4D-Intelli

single minutes interval, signTerms for tfidf

Significance filtering

命令提示符 - python

```
>>>
>>> signTerms = stat_CortfidfIR.query('Pvalue < 0.05')
>>> signTerms
          keyterms  correlations     Fvalue    Pvalue
14           abrupt      0.072909   9.542712  0.002124
19           absorb      0.036242   4.135755  0.042536
47         accounts     -0.047568   3.910518  0.048556
58      acknowledge      0.022526   4.266509  0.039407
78         addendum     -0.004348   6.700484  0.009931
88       adjournment     -0.022219   5.322804  0.021473
94    administration     -0.028975   4.773067  0.029391
104         adverse     -0.045732   4.575639  0.032934
106          advice     -0.018867   5.188072  0.023181
134           agent     -0.018044   5.147255  0.023726
142       agreement     -0.007181   7.496801  0.006411
156            alan      0.033705   5.916977  0.015360
162          alfred      0.069269   5.513721  0.019273
209         andersen      0.058650   5.078344  0.024677
223          angell     -0.034176   4.582217  0.032809
240       anticipate     -0.020726   4.143916  0.042333
246        anxieties      0.079805   6.583966  0.010593
252        apartment     -0.026787   4.612661  0.032237
269         applying     -0.012122   4.075300  0.044070
270          appoint     -0.001589   4.801393  0.028917
297        argentina     -0.043150   3.969108  0.046908
306       arithmetic      0.019944   5.075134  0.024722
313           arrest     -0.061127   4.240217  0.040016
317          article     -0.063873   4.312559  0.038363
328           ashton     -0.018461   4.052742  0.044657
339            asset     -0.039252   4.578618  0.032877
```

4D-Intelli

single minutes interval, signTerms for tfidf

Significance filtering

命令提示符 - python

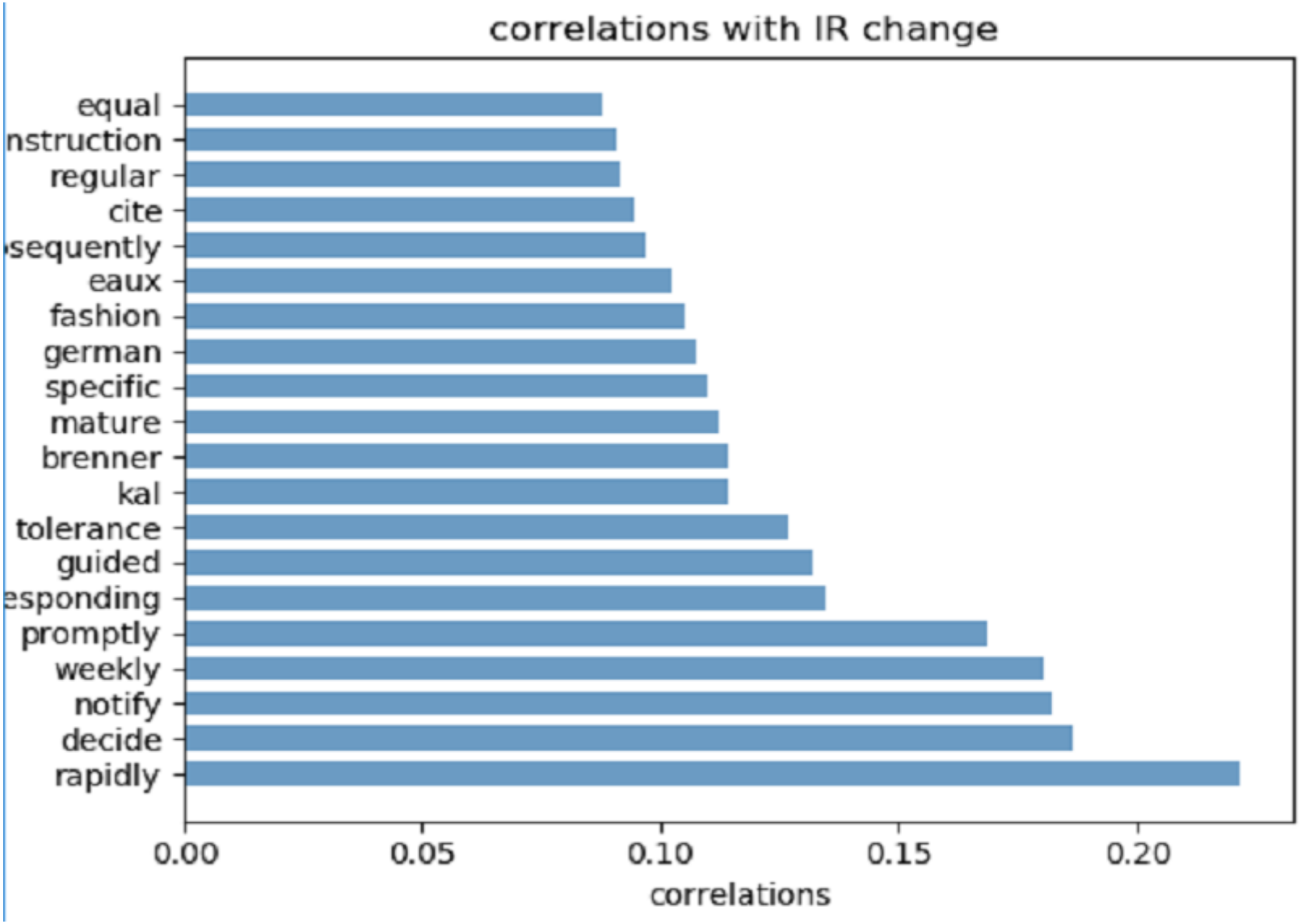
5473	unutilized	0.029730	4.249147	0.039808
5491	upturn	-0.044493	3.957876	0.047220
5505	usefully	-0.012655	5.141972	0.023798
5507	user	0.031460	7.196626	0.007556
5508	uso	0.000302	4.523661	0.033939
5516	utility	0.025373	4.901845	0.027297
5524	valerie	0.001910	4.121948	0.042881
5568	views	0.031827	4.157226	0.042005
5575	vincent	0.025928	6.324701	0.012232
5578	vis	-0.010367	4.840629	0.028272
5592	vor	0.012188	4.022287	0.045464
5599	wa	0.030266	5.599564	0.018361
5600	wag	0.045590	4.871449	0.027777
5608	wallace	0.040313	8.891278	0.003011
5621	warm	0.011740	12.686460	0.000405
5623	warning	0.011729	4.873856	0.027739
5628	wase	0.019924	4.058948	0.044495
5634	weak	-0.089707	9.803539	0.001848
5639	weakness	-0.082999	14.298548	0.000176
5641	weather	0.029195	5.411244	0.020423
5647	weekly	0.180152	9.012084	0.002822
5651	weigh	-0.039363	5.398709	0.020568
5705	wonna	-0.104112	3.878807	0.049474
5708	wording	0.010148	3.953679	0.047336
5723	worthy	-0.047582	6.072661	0.014078
5724	wright	0.018008	5.333638	0.021341
5734	yad	-0.016500	7.450588	0.006575
5753	zina	-0.045072	4.379574	0.036897

[417 rows x 4 columns]

4D-Intelli

single minutes interval, signTerms for tfidf

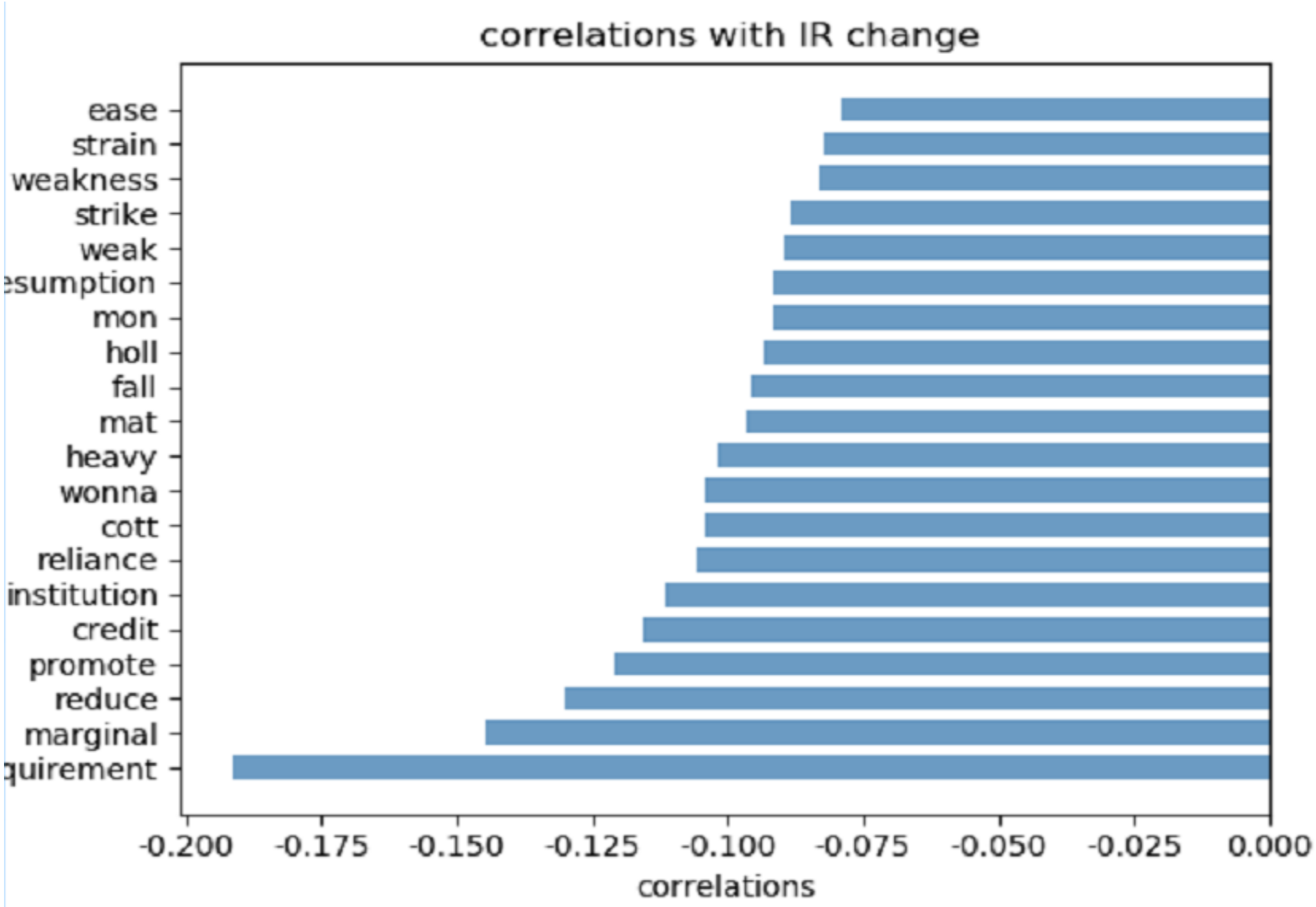
Correlation



4D-Intelli

single minutes interval, signTerms for tfidf

Correlation



4D-Intelli

single minutes interval, signTerms for tfidf

Model selection

	train_score	test_score	model
0	0.830728006	0.356506	SVR
1	0.932467532	0.814433	SVC
2	-3857178400	-1.4E+09	SGD
3	0.593098832	0.383987	BAYES
4	0	-0.01623	LL
5	0.699369713	-0.02541	ARD
6	0.505046557	-0.36883	PA
7	0.991313642	-4726.59	TS
8	0.991313642	-4728.46	L

single minutes interval, signTerms for tfidf

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_time	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.044957	0.008292	0.023428	0.003171	0.1	linear	{'C': 0.1, 'k	0.682171	0.658915	0.637795	0.65974	0.018102	9	0.992188	0.980469	0.988372	0.9870
3	1	0.051534	0.006026	0.023829	0.007179	0.1	rbf	{'C': 0.1, 'k	0.503876	0.503876	0.559055	0.522078	0.025943	18	0.503906	0.503906	0.635659	0.5478
4	2	0.033446	0.008894	0.014883	0.003902	0.5	linear	{'C': 0.5, 'k	0.658915	0.666667	0.653543	0.65974	0.005382	9	0.996094	0.996094	1	0.9973
5	3	0.051591	0.004959	0.021253	0.004992	0.5	rbf	{'C': 0.5, 'k	0.72093	0.689922	0.653543	0.688312	0.027498	8	0.886719	0.832031	0.848837	0.8558
6	4	0.038315	0.006248	0.007626	0.000334	0.8	linear	{'C': 0.8, 'k	0.635659	0.666667	0.653543	0.651948	0.012741	16	1	0.996094	1	0.9986
7	5	0.053924	0.003156	0.023465	0.004964	0.8	rbf	{'C': 0.8, 'k	0.751938	0.713178	0.677165	0.714286	0.030495	7	0.9375	0.878906	0.887597	0.9013
8	6	0.038121	0.010037	0.008205	0.004904	1	linear	{'C': 1, 'kerı	0.651163	0.658915	0.653543	0.654545	0.00325	12	1	1	1	
9	7	0.064391	0.015699	0.028635	0.005119	1	rbf	{'C': 1, 'kerı	0.751938	0.72093	0.685039	0.719481	0.027294	6	0.933594	0.894531	0.910853	0.9129
10	8	0.031367	0.006069	0.01389	0.001319	1.2	linear	{'C': 1.2, 'k	0.643411	0.658915	0.653543	0.651948	0.006444	16	1	1	1	
11	9	0.063803	0.0133	0.031318	0.007904	1.2	rbf	{'C': 1.2, 'k	0.744186	0.728682	0.708661	0.727273	0.014518	2	0.941406	0.914063	0.926357	0.9272
12	10	0.040302	0.015182	0.017208	0.003445	1.5	linear	{'C': 1.5, 'k	0.643411	0.674419	0.653543	0.657143	0.012941	11	1	1	1	
13	11	0.069033	0.014749	0.032504	0.015669	1.5	rbf	{'C': 1.5, 'k	0.751938	0.72093	0.700787	0.724675	0.021023	4	0.957031	0.9375	0.930233	0.9415
14	12	0.029065	0.008166	0.012476	0.001755	2	linear	{'C': 2, 'kerı	0.643411	0.666667	0.653543	0.654545	0.009545	12	1	1	1	
15	13	0.042622	0.00516	0.022434	0.001417	2	rbf	{'C': 2, 'kerı	0.744186	0.728682	0.700787	0.724675	0.017921	4	0.972656	0.964844	0.94186	0.9597
16	14	0.031696	0.008871	0.008135	0.004341	5	linear	{'C': 5, 'kerı	0.643411	0.666667	0.653543	0.654545	0.009545	12	1	1	1	
17	15	0.051673	0.001131	0.018275	0.004977	5	rbf	{'C': 5, 'kerı	0.736434	0.744186	0.708661	0.72987	0.015215	1	0.984375	0.972656	0.98062	0.9792
18	16	0.037401	0.005572	0.013755	0.004436	10	linear	{'C': 10, 'ke	0.643411	0.666667	0.653543	0.654545	0.009545	12	1	1	1	
19	17	0.043625	0.001386	0.021811	0.001748	10	rbf	{'C': 10, 'ke	0.728682	0.744186	0.708661	0.727273	0.014518	2	0.996094	0.992188	0.992248	0.993
20																		
21																		
22																		

best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
0.7422680412371134
>>>
```

4D-Intelli

year minutes interval, signTerms for bow

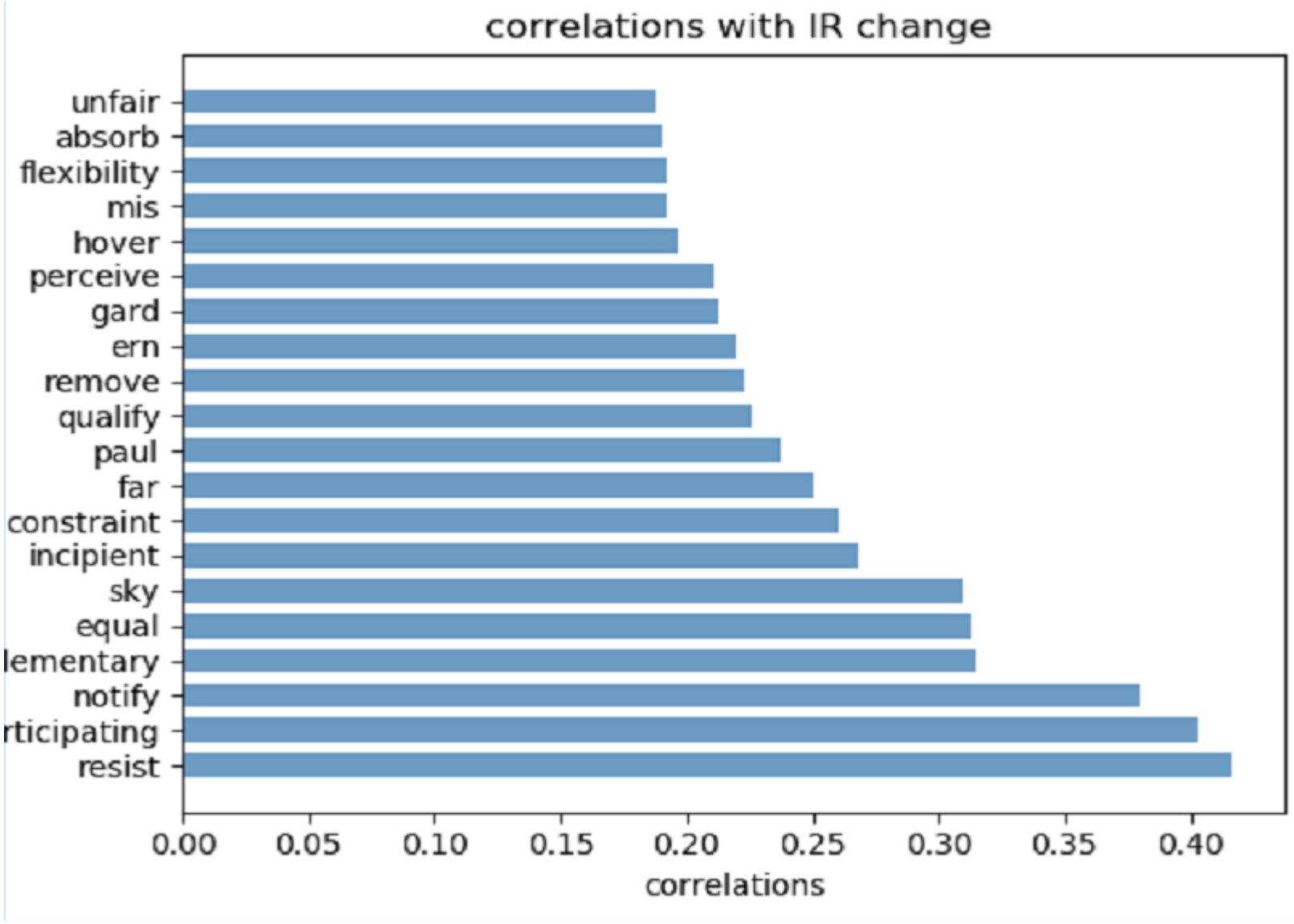
Correlation

```
bow_IR_diff = bow_IR.dropna()
bow_IR_year = bow_IR_diff.groupby(['year'],as_index=False).mean()


bow_IR_year['rateChange'] = bow_IR_year['fedRate'].shift(-1) - bow_IR_year['fedRate']
```

year minutes interval, signTerms for bow

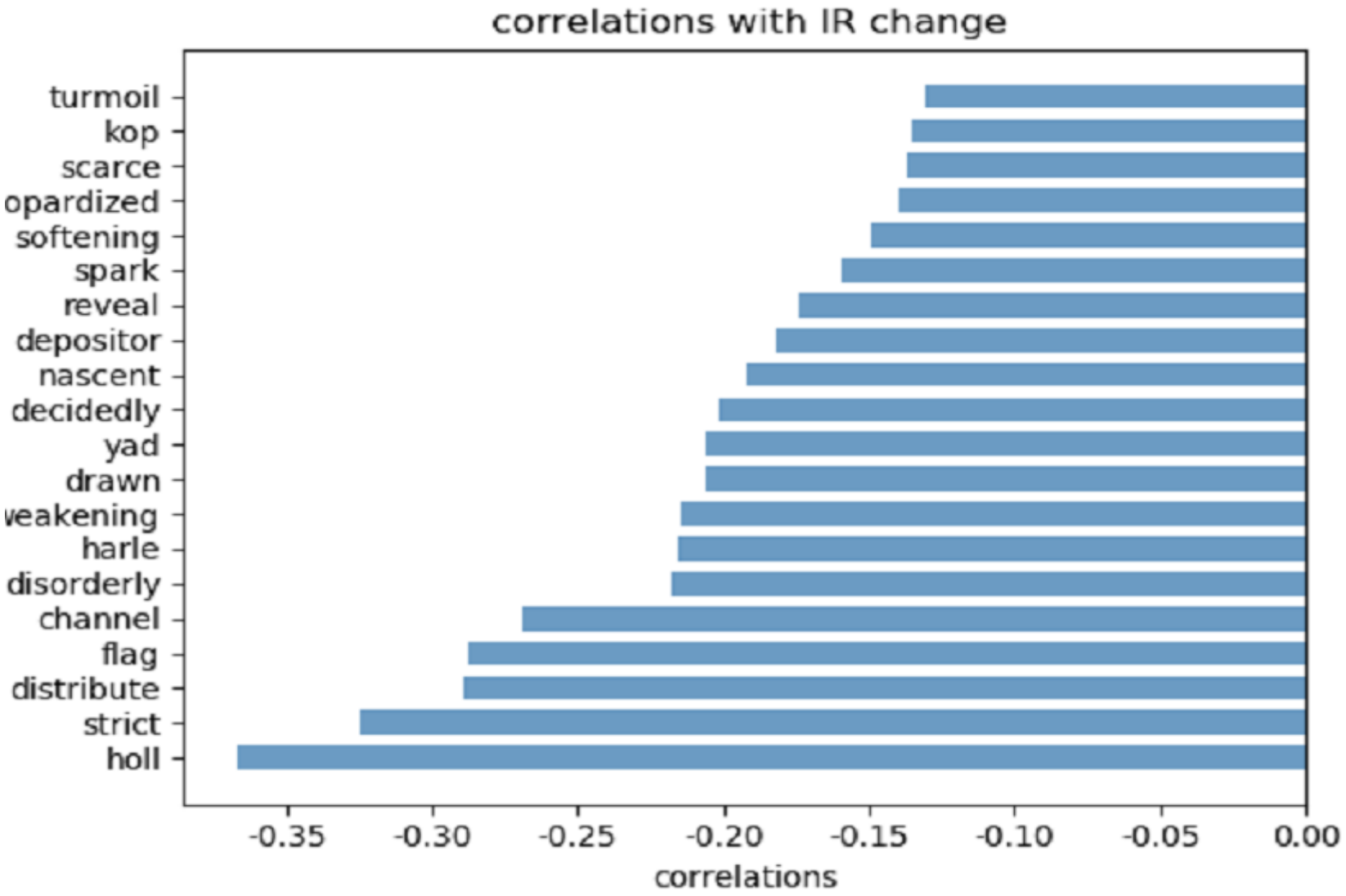
Correlation



4D-Intelli

year minutes interval, signTerms for bow

Correlation



4D-Intelli

year minutes interval, signTerms for bow

Model Selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_time	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.001335	0.001251	0.00034	0.000481	0.1	linear	{'C': 0.1, 'k	0.857143	0.692308	0.923077	0.825	0.09597	1	0.961538	1	0.925926	0.9624
3	1	0.004505	0.001875	0.000998	1.35E-06	0.1	rbf	{'C': 0.1, 'k	0.571429	0.538462	0.538462	0.55	0.015724	18	0.538462	0.555556	0.555556	0.5498
4	2	0	0	0	0	0.5	linear	{'C': 0.5, 'k	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
5	3	0.005236	0.001896	0.001153	0.000852	0.5	rbf	{'C': 0.5, 'k	0.714286	0.769231	0.538462	0.675	0.09739	13	0.769231	0.851852	0.555556	0.7255
6	4	0	0	0.003928	0.005556	0.8	linear	{'C': 0.8, 'k	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
7	5	0.005308	0.002803	0.001802	0.000482	0.8	rbf	{'C': 0.8, 'k	0.571429	0.615385	0.538462	0.575	0.031119	16	0.692308	0.777778	0.555556	0.6752
8	6	0.005527	0.004729	0	0	1	linear	{'C': 1, 'ker	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
9	7	0.005121	0.000171	0.001335	0.000467	1	rbf	{'C': 1, 'ker	0.571429	0.615385	0.538462	0.575	0.031119	16	0.692308	0.666667	0.555556	0.6381
10	8	0.001531	0.002165	0.005052	0.002999	1.2	linear	{'C': 1.2, 'k	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
11	9	0.006149	0.001126	0.001697	0.00095	1.2	rbf	{'C': 1.2, 'k	0.571429	0.615385	0.769231	0.65	0.084678	14	0.730769	0.740741	0.925926	0.7991
12	10	0.001931	0.001533	0.000542	0.000767	1.5	linear	{'C': 1.5, 'k	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
13	11	0.004968	0.001341	0.002017	0.000155	1.5	rbf	{'C': 1.5, 'k	0.571429	0.615385	0.769231	0.65	0.084678	14	0.769231	0.777778	0.814815	0.7872
14	12	0.004392	0.004346	0.000666	0.000471	2	linear	{'C': 2, 'ker	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
15	13	0.003593	0.000187	0.001682	0.000487	2	rbf	{'C': 2, 'ker	0.642857	0.692308	0.769231	0.7	0.052152	12	0.769231	0.814815	0.777778	0.7872
16	14	0.000342	0.000483	0.002829	0.003146	5	linear	{'C': 5, 'ker	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
17	15	0.002862	0.00027	0.001097	0.000143	5	rbf	{'C': 5, 'ker	0.785714	0.692308	0.846154	0.775	0.062514	3	0.769231	0.925926	0.777778	0.8243
18	16	0.000382	0.000541	0.000333	0.000471	10	linear	{'C': 10, 'ke	0.642857	0.692308	0.846154	0.725	0.086483	4	1	1	1	
19	17	0.002704	0.000893	0.001013	1.16E-05	10	rbf	{'C': 10, 'ke	0.857143	0.692308	0.846154	0.8	0.074863	2	0.884615	1	0.851852	0.9121
20																		
21																		
22																		

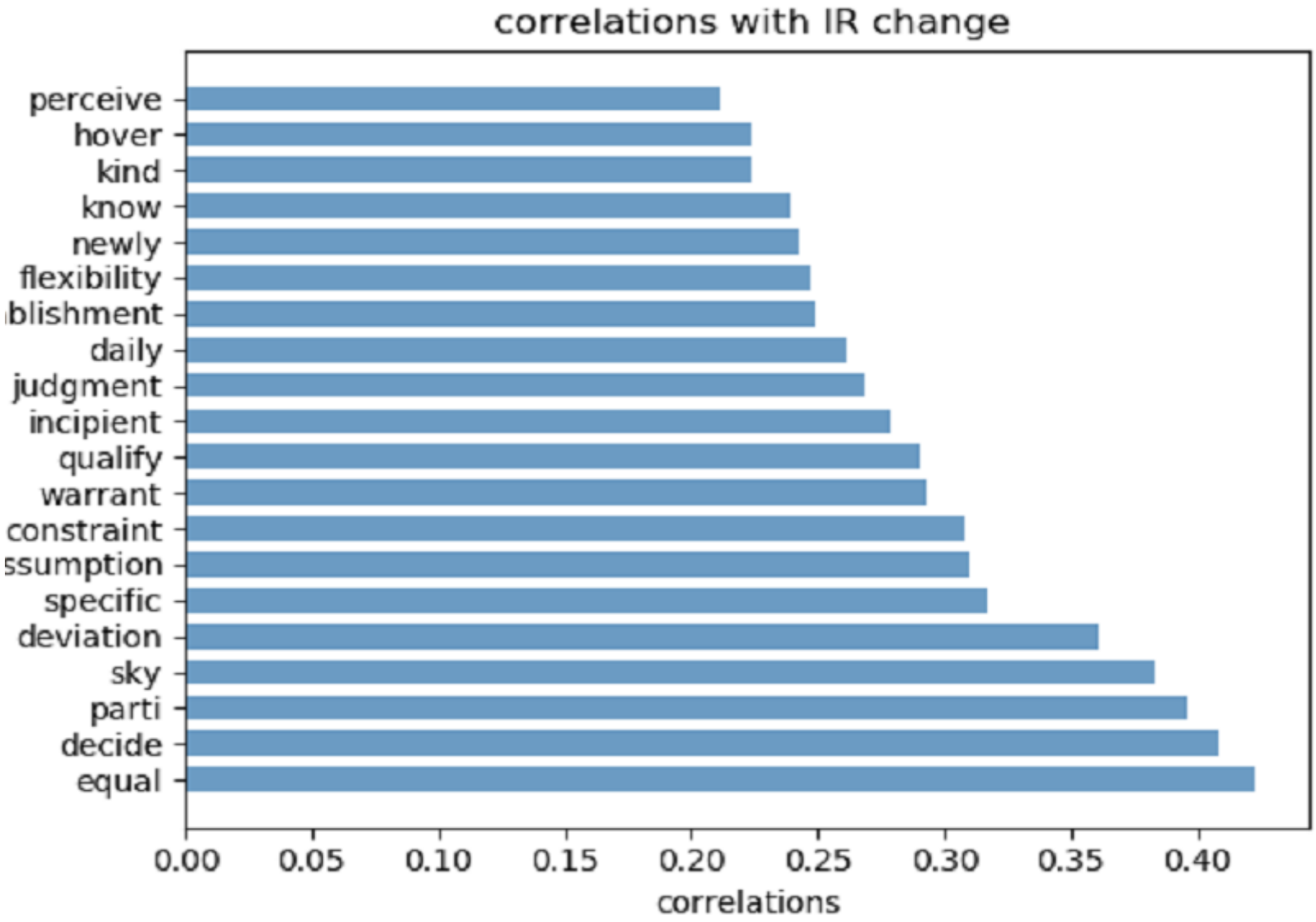
best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
0.9090909090909091
>>>
```

4D-Intelli

year minutes interval, signTerms for tfidf

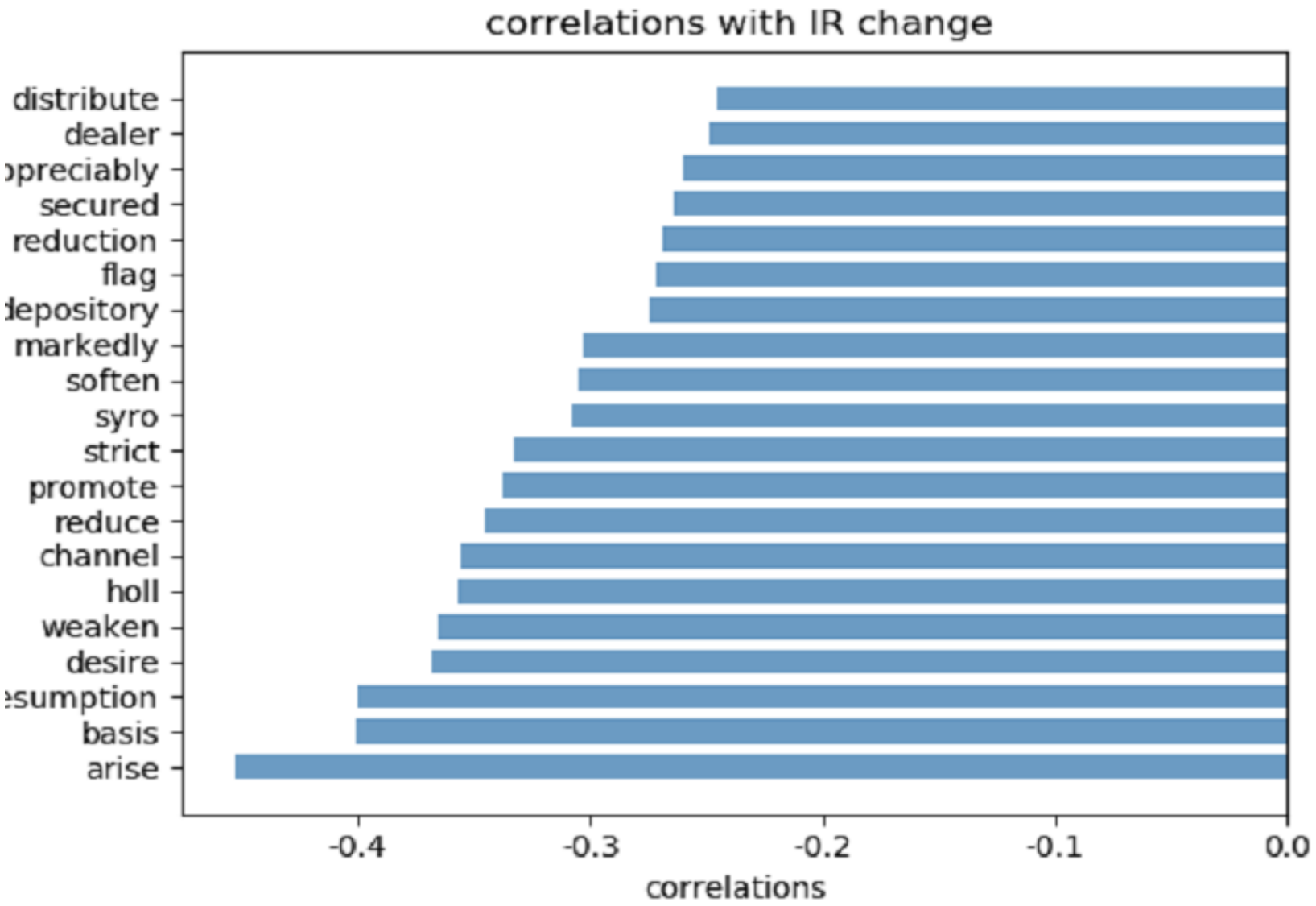
Correlation



4D-Intelli

year minutes interval, signTerms for bow

Correlation



4D-Intelli

year minutes interval, signTerms for bow

Model Selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_tim	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.000344	0.000487	0	0	0.1	linear	{'C': 0.1, 'k	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
3	1	0.004145	0.005172	0.000332	0.000469	0.1	rbf	{'C': 0.1, 'k	0.571429	0.538462	0.538462	0.55	0.015724	18	0.538462	0.555556	0.555556	0.5498
4	2	0.00128	0.001811	0.003492	0.004939	0.5	linear	{'C': 0.5, 'k	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
5	3	0.00534	0.004135	0.005731	0.005023	0.5	rbf	{'C': 0.5, 'k	0.571429	0.615385	0.538462	0.575	0.031119	17	0.653846	0.62963	0.555556	0.613
6	4	0.003692	0.004532	0	0	0.8	linear	{'C': 0.8, 'k	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
7	5	0.001255	0.001775	0.004815	0.00489	0.8	rbf	{'C': 0.8, 'k	0.642857	0.692308	0.615385	0.65	0.031449	16	0.807692	0.962963	0.592593	0.7877
8	6	0.001677	0.002372	0.000334	0.000473	1	linear	{'C': 1, 'kerı	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
9	7	0.004761	0.006038	0	0	1	rbf	{'C': 1, 'kerı	0.714286	0.692308	0.692308	0.7	0.010483	15	1	0.962963	0.740741	0.9012
10	8	0	0	0	0	1.2	linear	{'C': 1.2, 'k	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
11	9	0.002494	0.001902	0	0	1.2	rbf	{'C': 1.2, 'k	0.928571	0.769231	0.769231	0.825	0.076001	14	1	1	0.777778	0.9259
12	10	0.000866	0.001224	0	0	1.5	linear	{'C': 1.5, 'k	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
13	11	0.002001	0.002143	0.001528	0.002161	1.5	rbf	{'C': 1.5, 'k	0.857143	0.923077	0.923077	0.9	0.031449	2	1	1	1	
14	12	0.005507	0.004599	0.003744	0.005294	2	linear	{'C': 2, 'kerı	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
15	13	0.003359	0.00475	0.007778	0.004322	2	rbf	{'C': 2, 'kerı	0.785714	0.923077	0.923077	0.875	0.065518	3	1	1	1	
16	14	0.005716	0.00374	0.000573	0.000811	5	linear	{'C': 5, 'kerı	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
17	15	0.009953	0.0065	0.000891	0.00126	5	rbf	{'C': 5, 'kerı	0.785714	0.846154	1	0.875	0.090215	3	1	1	1	
18	16	0.00425	0.000709	0.001901	0.000419	10	linear	{'C': 10, 'ke	0.785714	0.769231	1	0.85	0.104303	5	1	1	1	
19	17	0.001693	0.002394	0.004059	0.005054	10	rbf	{'C': 10, 'ke	0.785714	1	1	0.925	0.102208	1	1	1	1	
20																		
21																		
22																		

best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
1.0
>>>
```

4D-Intelli

single interval, signTerms for bow, real estate

Data source

Wilshire US Real Estate Investment Trust Total Market Index (Wilshire US REIT)

Index, Daily, Not Seasonally Adjusted — 1977-12-30 to 2019-03-12 (19 hours ago)

Wilshire US Real Estate Securities Price Index (Wilshire US RESI)

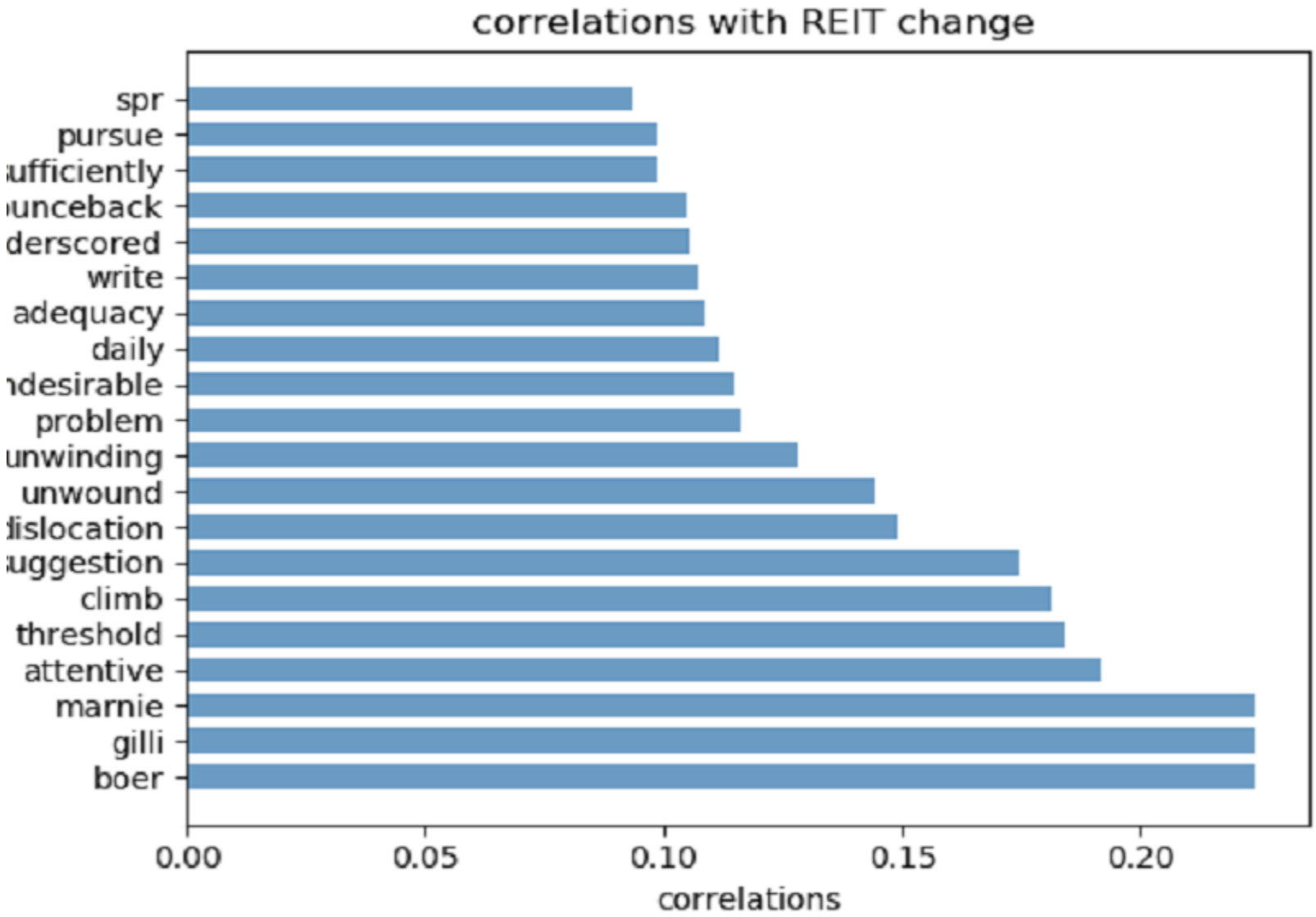
Wilshire US Real Estate Securities Total Market Index (Wilshire US RESI)

Index, Daily, Not Seasonally Adjusted — 1977-12-30 to 2019-03-12 (19 hours ago)

4D-Intelli

single interval, signTerms for bow, real estate

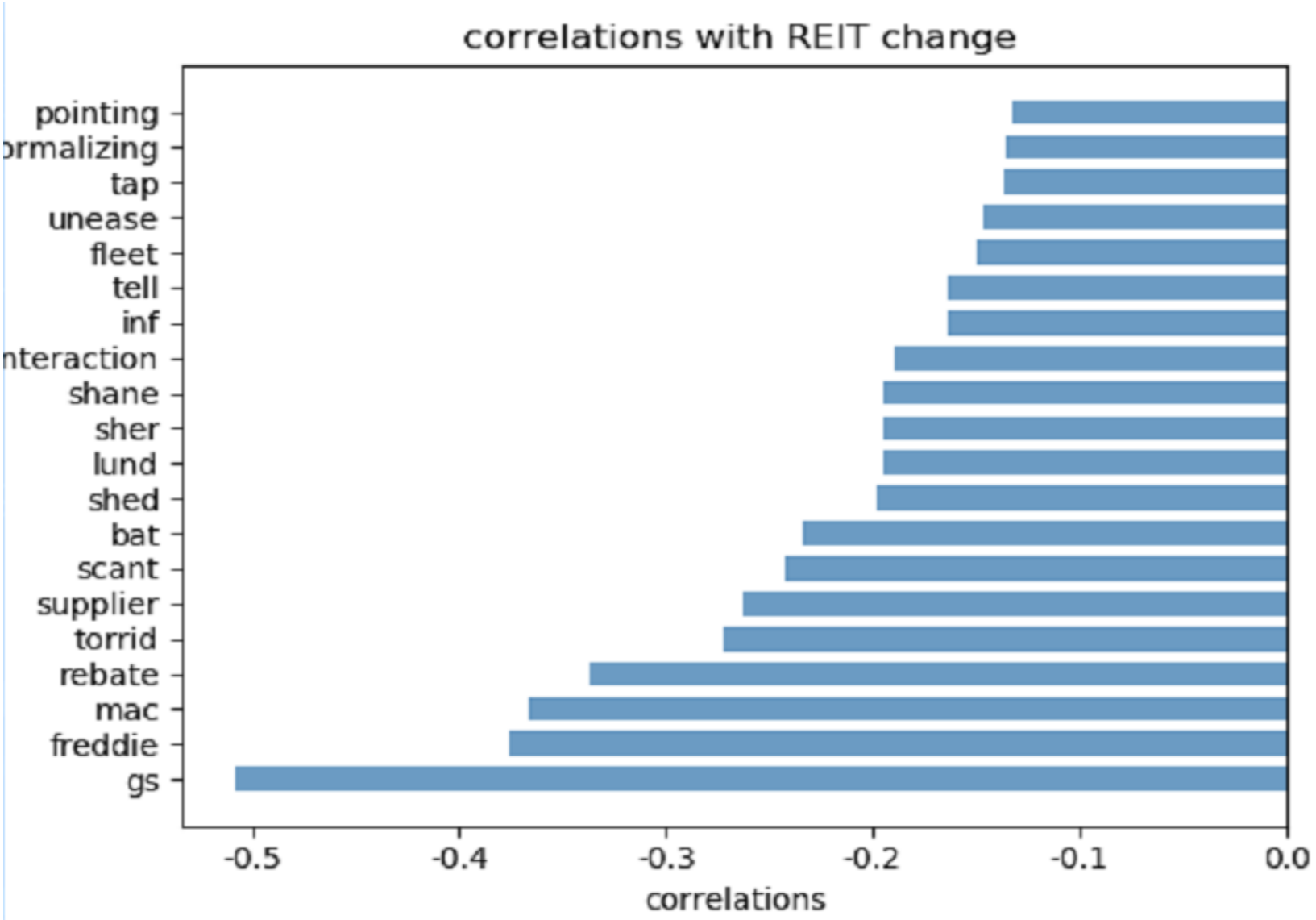
Correlation



4D-Intelli

single interval, signTerms for bow, real estate

Correlation



4D-Intelli

single interval, signTerms for bow, real estate

Model selection

	train_score	test_score	model
0	0.646735	0.182921	SVR
1	0.867424	0.835821	SVC
2	0.469581	0.089544	SGD
3	0.517452	0.362774	BAYES
4	0	-0.03922	LL
5	0.584756	0.187864	ARD
6	-0.52009	-2.98837	PA
7	0.218981	-2.31309	TS
8	0.681382	-14.6919	L

single interval, signTerms for bow, real estate

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_tim	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.015076	0.001962	0.006009	0.001575	0.1	linear	{'C': 0.1, 'k	0.629213	0.761364	0.747126	0.712121	0.059408	8	0.931429	0.875	0.892655	0.8996
3	1	0.020403	0.003579	0.006074	0.004647	0.1	rbf	{'C': 0.1, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	18	0.594286	0.596591	0.59322	0.5946
4	2	0.01184	0.002772	0.005404	0.003972	0.5	linear	{'C': 0.5, 'k	0.58427	0.704545	0.678161	0.655303	0.051783	10	0.942857	0.869318	0.903955	0.9053
5	3	0.020403	0.003327	0.008471	0.001109	0.5	rbf	{'C': 0.5, 'k	0.707865	0.613636	0.609195	0.643939	0.045624	17	0.834286	0.8125	0.79096	0.8125
6	4	0.012283	0.003785	0.003507	0.001864	0.8	linear	{'C': 0.8, 'k	0.617978	0.670455	0.678161	0.655303	0.026803	10	0.942857	0.897727	0.932203	0.9242
7	5	0.017736	0.001529	0.008881	0.001445	0.8	rbf	{'C': 0.8, 'k	0.707865	0.784091	0.770115	0.753788	0.03324	1	0.868571	0.857955	0.841808	0.8561
8	6	0.01176	0.003364	0.006353	0.001557	1	linear	{'C': 1, 'ker	0.629213	0.681818	0.689655	0.666667	0.026899	9	0.937143	0.903409	0.932203	0.9242
9	7	0.01583	0.002313	0.009091	0.001213	1	rbf	{'C': 1, 'ker	0.707865	0.772727	0.770115	0.75	0.030067	2	0.868571	0.869318	0.841808	0.8598
10	8	0.012861	0.002604	0.006272	0.002788	1.2	linear	{'C': 1.2, 'k	0.617978	0.659091	0.689655	0.655303	0.029383	10	0.942857	0.903409	0.932203	0.9261
11	9	0.020659	0.002103	0.006637	0.001431	1.2	rbf	{'C': 1.2, 'k	0.696629	0.772727	0.781609	0.75	0.038232	2	0.88	0.875	0.841808	0.8656
12	10	0.014366	0.002474	0.004066	0.000556	1.5	linear	{'C': 1.5, 'k	0.606742	0.647727	0.701149	0.651515	0.038632	13	0.942857	0.903409	0.932203	0.9261
13	11	0.020779	0.004044	0.005267	0.003037	1.5	rbf	{'C': 1.5, 'k	0.696629	0.784091	0.770115	0.75	0.038484	2	0.88	0.875	0.847458	0.8674
14	12	0.012067	0.000481	0.001155	0.000191	2	linear	{'C': 2, 'ker	0.606742	0.647727	0.701149	0.651515	0.038632	13	0.942857	0.903409	0.932203	0.9261
15	13	0.016957	0.002243	0.009434	0.000869	2	rbf	{'C': 2, 'ker	0.685393	0.772727	0.770115	0.742424	0.040685	5	0.885714	0.875	0.853107	0.8712
16	14	0.013	0.003307	0.005938	0.003466	5	linear	{'C': 5, 'ker	0.606742	0.647727	0.689655	0.647727	0.033847	15	0.937143	0.903409	0.932203	0.9242
17	15	0.02542	0.005535	0.008125	0.001509	5	rbf	{'C': 5, 'ker	0.640449	0.761364	0.747126	0.715909	0.054125	7	0.931429	0.875	0.909605	0.9053
18	16	0.015272	0.000934	0.00396	0.002212	10	linear	{'C': 10, 'ke	0.606742	0.647727	0.689655	0.647727	0.033847	15	0.942857	0.903409	0.932203	0.9261
19	17	0.021278	0.000528	0.007923	0.000505	10	rbf	{'C': 10, 'ke	0.640449	0.75	0.770115	0.719697	0.057105	6	0.942857	0.880682	0.903955	0.9091
20																		
21																		
22																		

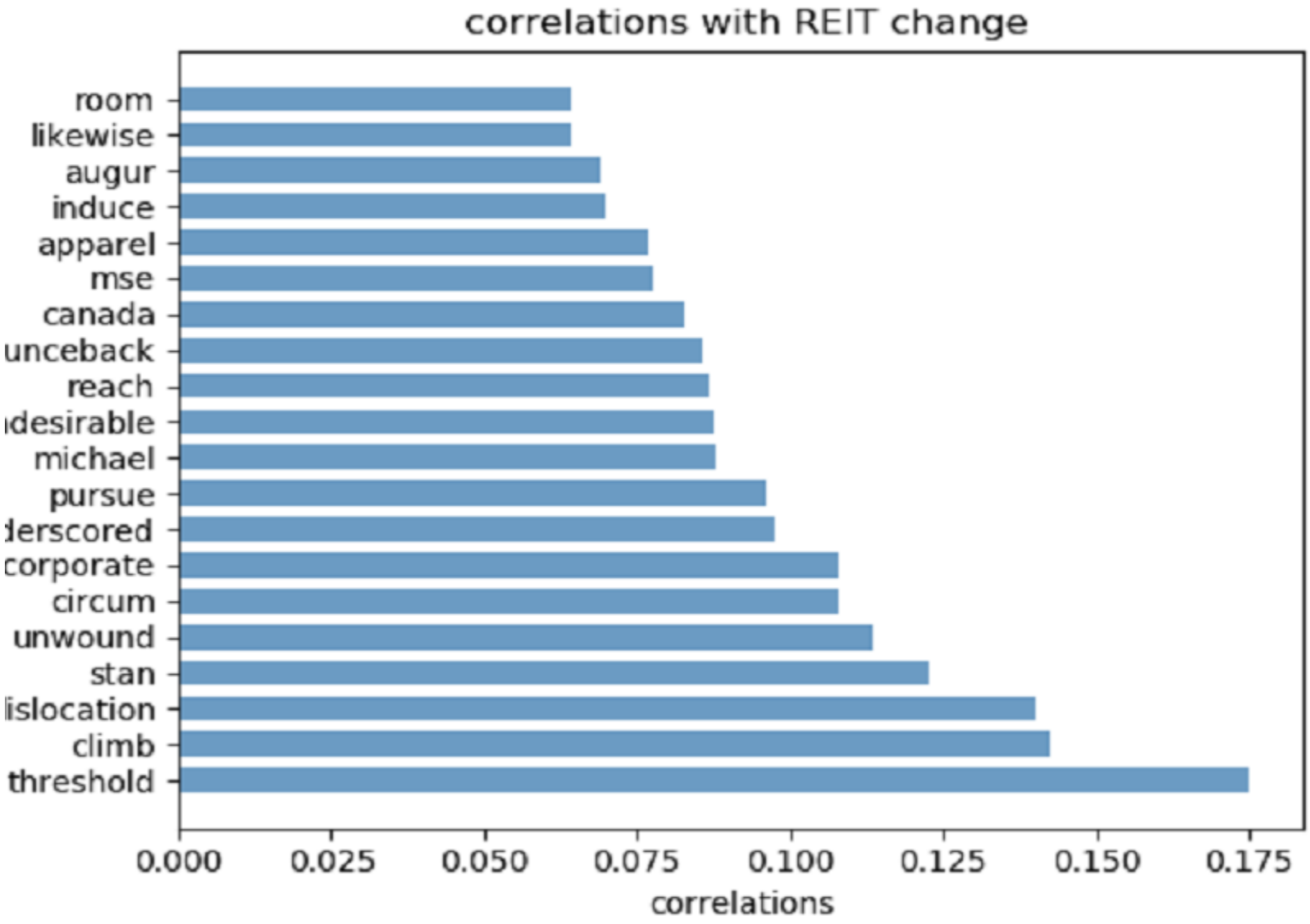
best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
0.8208955223880597
>>>
```

4D-Intelli

single interval, signTerms for tfidf, real estate

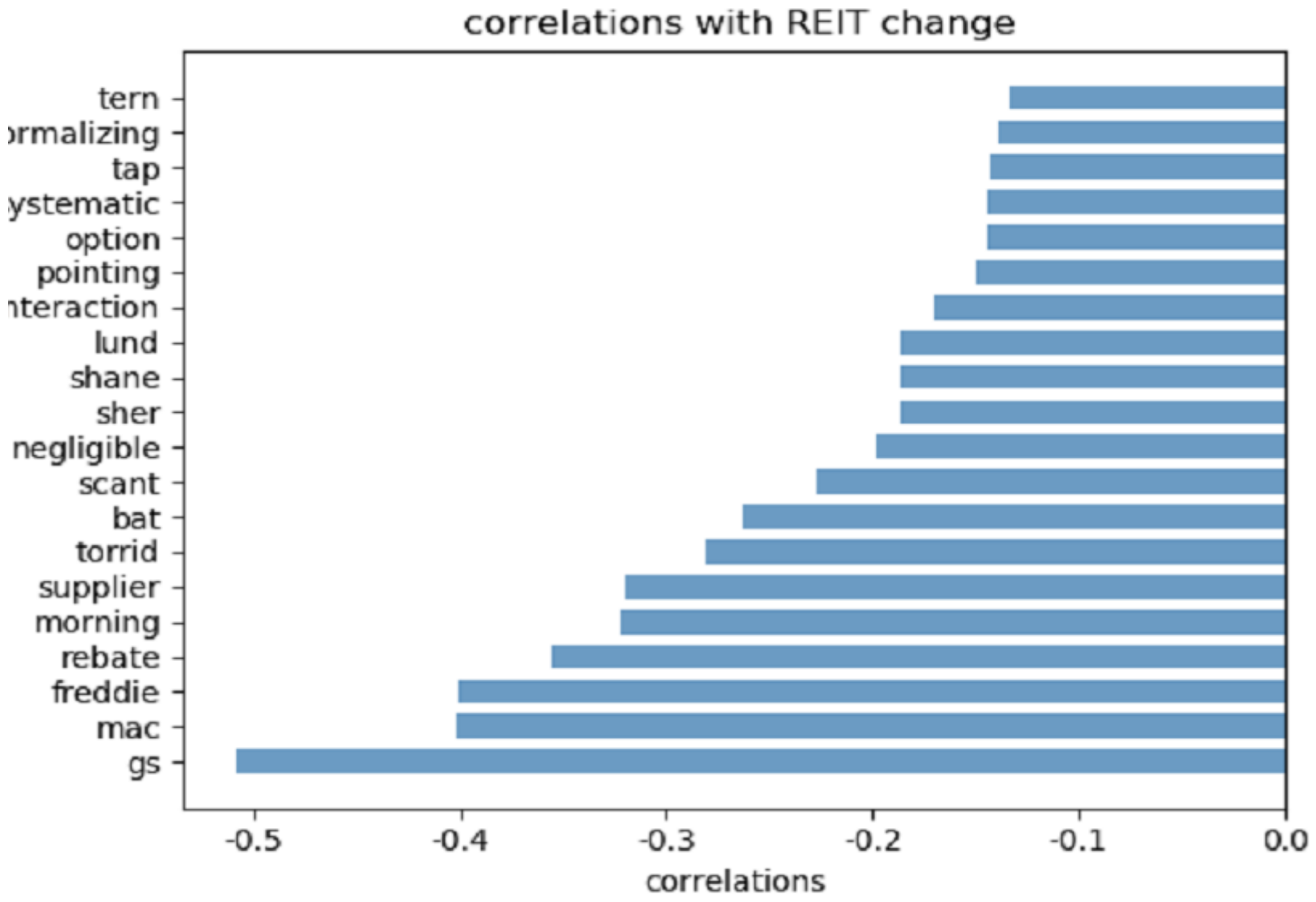
Correlation



4D-Intelli

single interval, signTerms for tfidf, real estate

Correlation



4D-Intelli

single interval, signTerms for tfidf, real estate

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_time	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.013695	0.002424	0.007534	0.005384	0.1	linear	{'C': 0.1, 'k	0.741573	0.715909	0.770115	0.742424	0.022074	9	0.948571	0.926136	0.915254	0.9299
3	1	0.015562	0.006295	0.007951	0.005624	0.1	rbf	{'C': 0.1, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	18	0.594286	0.596591	0.59322	0.5946
4	2	0.01182	0.00025	0	0	0.5	linear	{'C': 0.5, 'k	0.662921	0.681818	0.804598	0.715909	0.062658	15	0.954286	0.943182	0.932203	0.9432
5	3	0.012812	0.000889	0.008432	0.006299	0.5	rbf	{'C': 0.5, 'k	0.674157	0.772727	0.793103	0.746212	0.05205	8	0.862857	0.852273	0.841808	0.8523
6	4	0.010424	0.000596	0.000592	0.000837	0.8	linear	{'C': 0.8, 'k	0.662921	0.704545	0.804598	0.723485	0.059366	11	0.948571	0.943182	0.932203	0.9413
7	5	0.014968	0.005414	0.004111	0.005813	0.8	rbf	{'C': 0.8, 'k	0.707865	0.772727	0.793103	0.757576	0.036408	7	0.874286	0.869318	0.887006	0.876
8	6	0.014766	0.005789	0.007613	0.005387	1	linear	{'C': 1, 'kerı	0.662921	0.715909	0.816092	0.731061	0.063439	10	0.948571	0.943182	0.932203	0.9413
9	7	0.009552	0.004041	0.005061	0.004558	1	rbf	{'C': 1, 'kerı	0.719101	0.784091	0.804598	0.768939	0.036509	6	0.874286	0.869318	0.892655	0.8787
10	8	0.007732	0.005599	0.003372	0.004769	1.2	linear	{'C': 1.2, 'k	0.662921	0.704545	0.804598	0.723485	0.059366	11	0.954286	0.9375	0.932203	0.941
11	9	0.012552	0.00273	0	0	1.2	rbf	{'C': 1.2, 'k	0.741573	0.795455	0.804598	0.780303	0.02787	5	0.874286	0.869318	0.887006	0.876
12	10	0.007411	0.004541	0	0	1.5	linear	{'C': 1.5, 'k	0.662921	0.704545	0.793103	0.719697	0.054212	14	0.96	0.9375	0.932203	0.9432
13	11	0.012487	0.000141	0.011125	0.000846	1.5	rbf	{'C': 1.5, 'k	0.741573	0.818182	0.804598	0.787879	0.033482	4	0.891429	0.880682	0.903955	0.8920
14	12	0.008058	0.005741	0.003758	0.005314	2	linear	{'C': 2, 'kerı	0.662921	0.704545	0.804598	0.723485	0.059366	11	0.96	0.948864	0.932203	0.9470
15	13	0.014189	0.002564	0.011201	0.000616	2	rbf	{'C': 2, 'kerı	0.741573	0.852273	0.816092	0.80303	0.046236	1	0.902857	0.897727	0.915254	0.905
16	14	0.026671	0.014393	0.007101	0.005038	5	linear	{'C': 5, 'kerı	0.651685	0.704545	0.793103	0.715909	0.058286	15	0.954286	0.948864	0.937853	0.9470
17	15	0.015454	0.00592	0.008389	0.005978	5	rbf	{'C': 5, 'kerı	0.752809	0.784091	0.850575	0.795455	0.040711	2	0.925714	0.926136	0.915254	0.9223
18	16	0.022215	0.009257	0.00339	0.004795	10	linear	{'C': 10, 'ke	0.662921	0.693182	0.781609	0.712121	0.050268	17	0.954286	0.943182	0.937853	0.9451
19	17	0.012855	0.001932	0.003501	0.004951	10	rbf	{'C': 10, 'ke	0.752809	0.75	0.885057	0.795455	0.06283	2	0.948571	0.948864	0.920904	0.9394
20																		
21																		
22																		

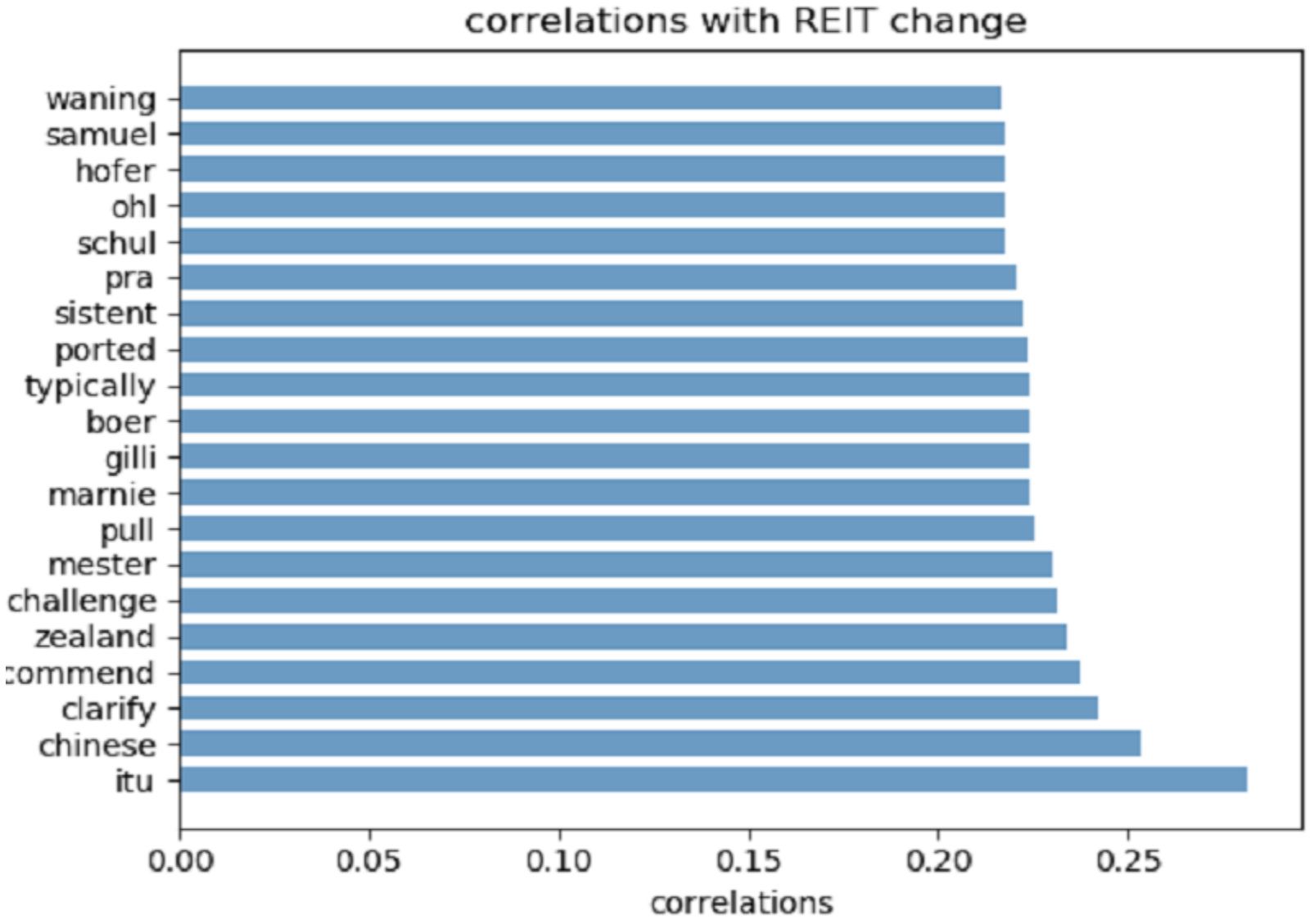
best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
0.8805970149253731
>>>
```

4D-Intelli

single minutes interval, LSA for tfidf, real estate

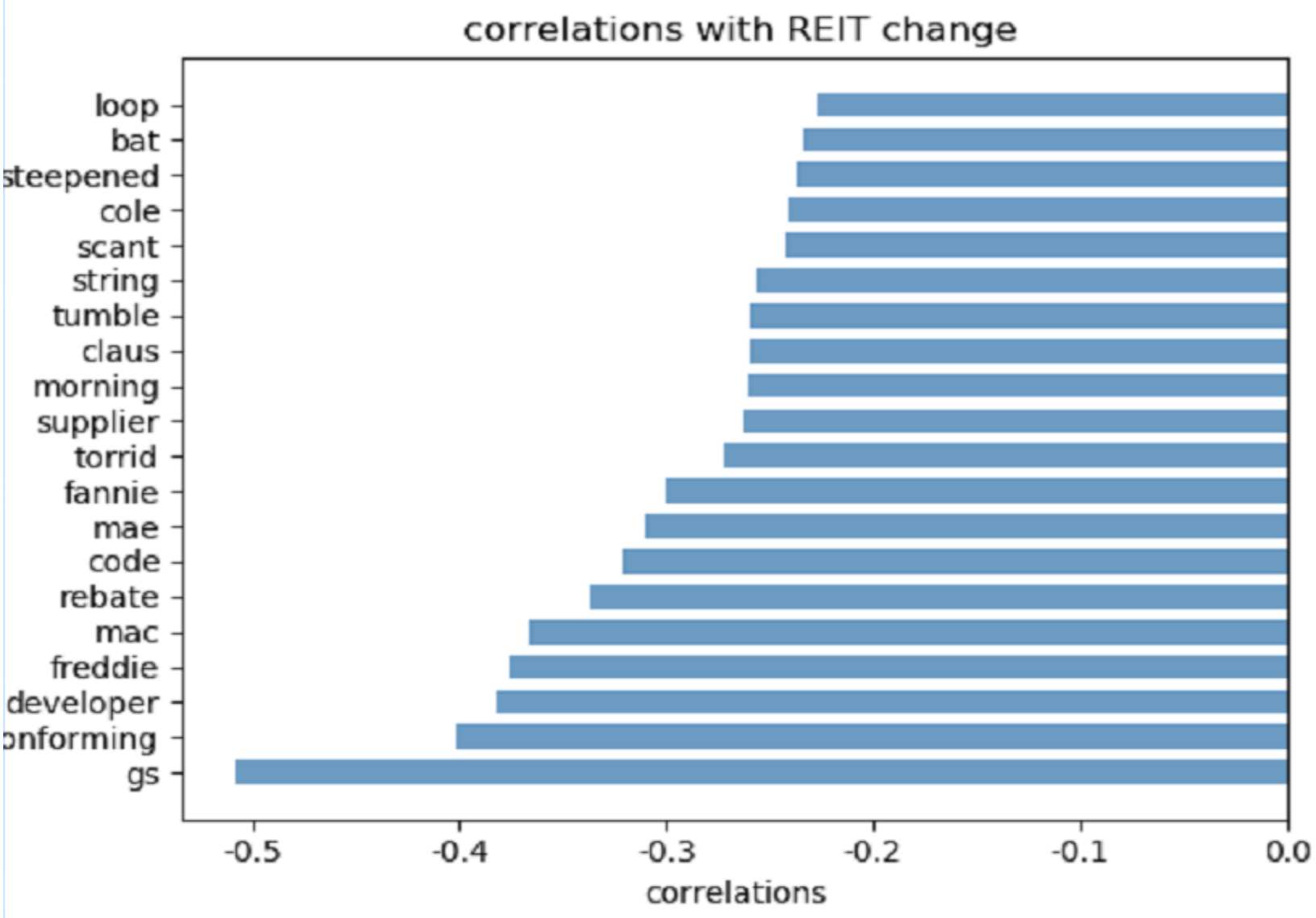
Correlation



4D-Intelli

single minutes interval, LSA for tfidf, real estate

Correlation



4D-Intelli

single minutes interval, LSA for tfidf, real estate

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_time	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0.002785	0.001886	0	0	0.1	linear	{'C': 0.1, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
3	1	0.006047	0.003585	0.002436	0.002252	0.1	rbf	{'C': 0.1, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
4	2	0.001017	0.001438	0	0	0.5	linear	{'C': 0.5, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
5	3	0.00542	0.003937	0.002513	0.003554	0.5	rbf	{'C': 0.5, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
6	4	0.003062	0.001601	0.002515	0.001448	0.8	linear	{'C': 0.8, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
7	5	0.008432	0.001679	0.003069	0.000585	0.8	rbf	{'C': 0.8, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
8	6	0.00598	0.000297	0.001051	0.000859	1	linear	{'C': 1, 'ker	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
9	7	0.006742	0.000613	0.00161	0.000837	1	rbf	{'C': 1, 'ker	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
10	8	0.004645	0.003062	0.001868	0.002642	1.2	linear	{'C': 1.2, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
11	9	0.007219	0.000837	0.002886	0.001958	1.2	rbf	{'C': 1.2, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
12	10	0.005067	0.003839	0.002126	0.003006	1.5	linear	{'C': 1.5, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
13	11	0.010816	0.000936	0.003263	0.000686	1.5	rbf	{'C': 1.5, 'k	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
14	12	0.004919	0.002177	0.001025	0.000847	2	linear	{'C': 2, 'ker	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.610169	0.6003
15	13	0.005932	0.00346	0.003328	0.002357	2	rbf	{'C': 2, 'ker	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
16	14	0.005323	0.000768	0.001392	0.001	5	linear	{'C': 5, 'ker	0.640449	0.647727	0.643678	0.643939	0.002985	2	0.645714	0.715909	0.768362	0.7099
17	15	0.005999	0.003378	0.002511	0.002877	5	rbf	{'C': 5, 'ker	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
18	16	0.003545	0.003086	0.000514	0.000726	10	linear	{'C': 10, 'ke	0.741573	0.693182	0.62069	0.685606	0.049637	1	0.708571	0.715909	0.80791	0.744
19	17	0.006689	0.001368	0.000688	0.000486	10	rbf	{'C': 10, 'ke	0.595506	0.590909	0.597701	0.594697	0.002824	3	0.594286	0.596591	0.59322	0.5946
20																		
21																		
22																		

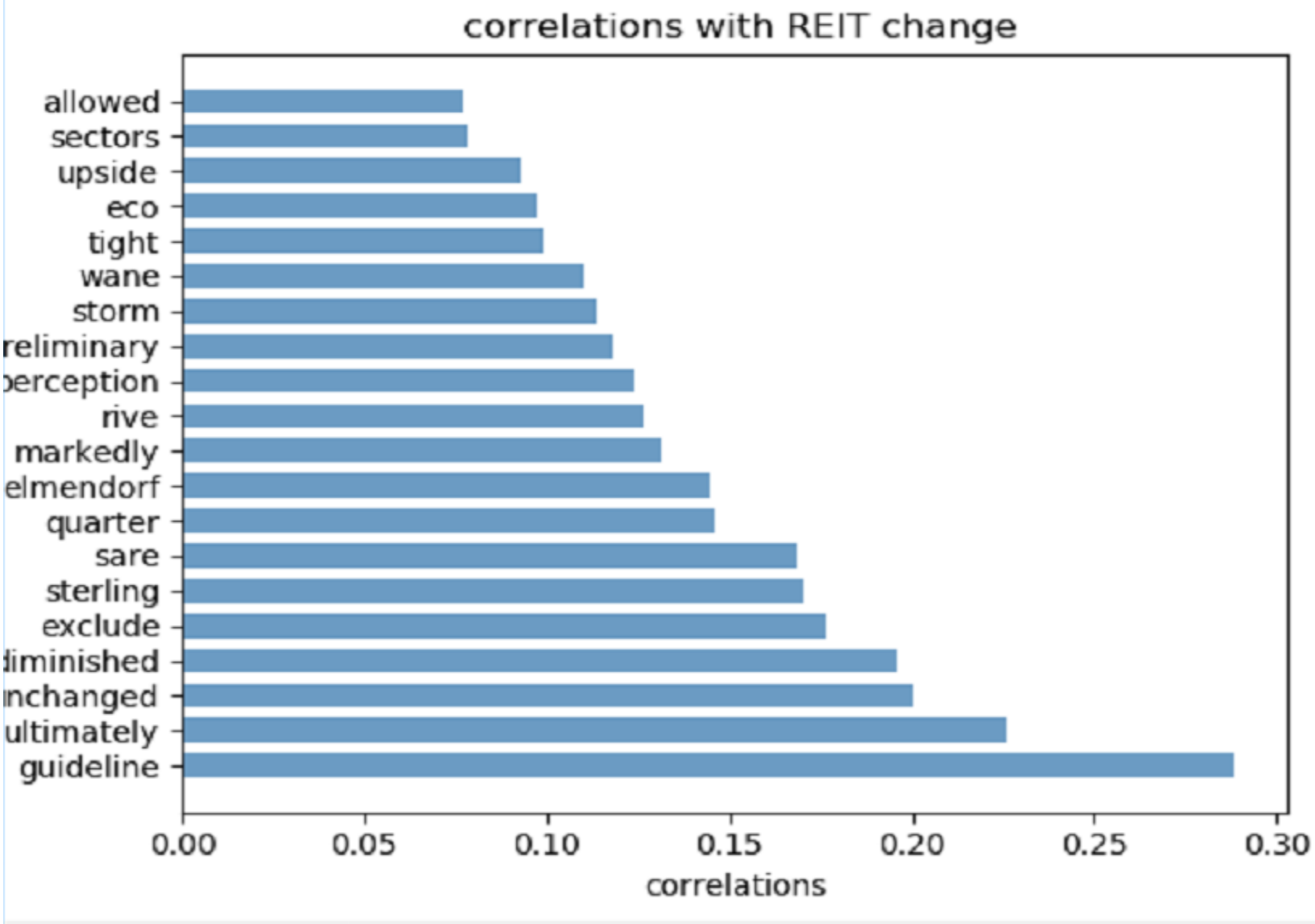
best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test,y_test)
0.746268656716418
>>>
```

4D-Intelli

year interval, signTerms for tfidf, real estate

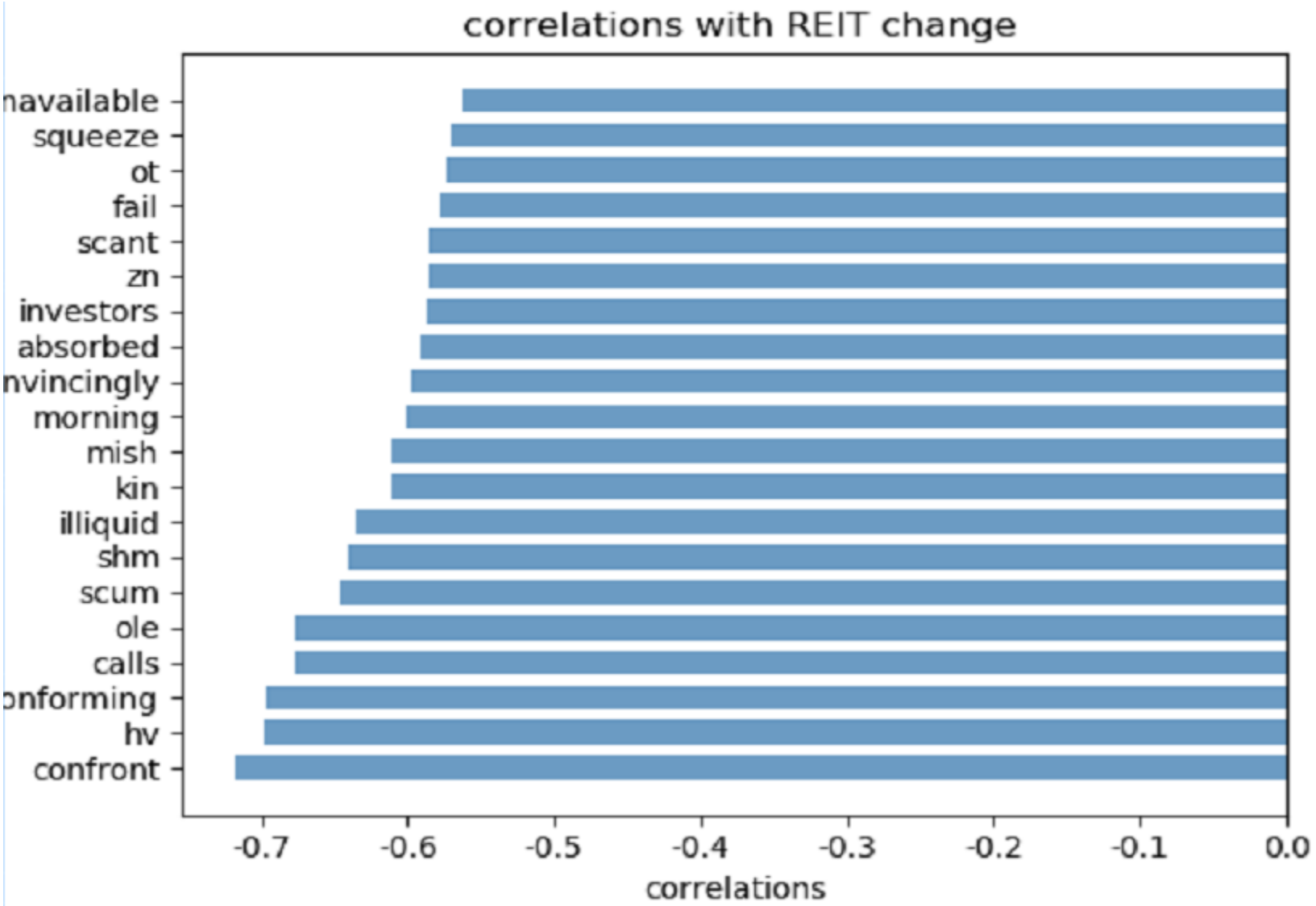
Correlation



4D-Intelli

year interval, signTerms for tfidf, real estate

Correlation



4D-Intelli

year interval, signTerms for tfidf, real estate

Model selection

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1		mean_fit_ti	std_fit_tim	mean_scor	std_score_t	param_C	param_ker	params	split0_test_	split1_test_	split2_test_	mean_test_	std_test_sc	rank_test_s	split0_train	split1_train	split2_train	mean_tr
2	0	0	0	0.003383	0.004785	0.1	linear	{'C': 0.1, 'k	0.833333	0.909091	0.9	0.878788	0.034551	7	1	1	1	
3	1	0.001111	0.001571	0.000332	0.00047	0.1	rbf	{'C': 0.1, 'k	0.833333	0.818182	0.9	0.848485	0.034551	9	0.857143	0.863636	0.826087	0.8489
4	2	0.004591	0.004241	0.000666	0.000471	0.5	linear	{'C': 0.5, 'k	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
5	3	0.001551	0.00112	0.000887	0.000672	0.5	rbf	{'C': 0.5, 'k	0.833333	0.818182	0.9	0.848485	0.034551	9	0.857143	0.954545	0.956522	0.9227
6	4	0.000929	0.001314	0	0	0.8	linear	{'C': 0.8, 'k	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
7	5	0.004379	0.006193	0.000343	0.000485	0.8	rbf	{'C': 0.8, 'k	0.833333	0.909091	0.9	0.878788	0.034551	7	0.952381	1	0.956522	0.9696
8	6	0.003775	0.004648	0.003741	0.004601	1	linear	{'C': 1, 'ker	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
9	7	0.002629	0.001859	0	0	1	rbf	{'C': 1, 'ker	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
10	8	0.001343	0.000951	0	0	1.2	linear	{'C': 1.2, 'k	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
11	9	0.002026	0.002865	0.00382	0.004722	1.2	rbf	{'C': 1.2, 'k	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
12	10	0.004868	0.004992	0.001586	0.001123	1.5	linear	{'C': 1.5, 'k	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
13	11	0.004644	0.004729	0.00034	0.000481	1.5	rbf	{'C': 1.5, 'k	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
14	12	0	0	0	0	2	linear	{'C': 2, 'ker	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
15	13	0.004472	0.004236	0.000666	0.000471	2	rbf	{'C': 2, 'ker	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
16	14	0.001153	0.001631	0.0003	0.000424	5	linear	{'C': 5, 'ker	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
17	15	0.00078	0.001103	0.00081	0.001145	5	rbf	{'C': 5, 'ker	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
18	16	0	0	0.003511	0.004965	10	linear	{'C': 10, 'ke	0.833333	0.909091	0.8	0.848485	0.044947	9	1	1	1	
19	17	0	0	0	0	10	rbf	{'C': 10, 'ke	0.833333	0.909091	1	0.909091	0.06776	1	0.952381	1	0.956522	0.9696
20																		
21																		
22																		

best2

```
>>> df_gs.to_csv('best2.csv')
>>> gs.score(X_test, y_test)
1.0
```

4D-Intelli

Tf-idf? tf-iwf!

Inverse document frequency (Wikipedia)

Inverse document frequency [edit]

The **inverse document frequency** is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

with

- N: total number of documents in the corpus $N = |D|$
- $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$.

4D-Intelli

Tf-idf? tf-iwf!

$$\text{Inverse word frequency} = \log \frac{\text{Total frequency showing up in all docs}}{\text{Total frequency showing up in one doc}}$$

Future Prospects

Progress Check

Improvement in Data Processing

Data Analysis

Future Prospects

4D-Intelli

Project Gantt

Gantt Chart – 5 weeks

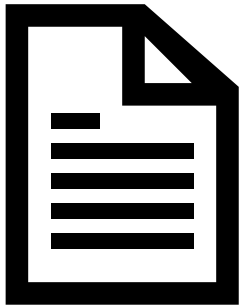
FOMA Minutes Timeline

As os 03/15/2019

Major Tasks	Start	Finish	Week 1	2	3	4	5
Initial Plan - Complete							
Identify Goals	15-Feb	23-Feb					
Sub-tasks	15-Feb	23-Feb					
Division of Work	15-Feb	23-Feb					
Experimental Process - Complete							
Step1: Web Scrapping	15-Feb	23-Feb					
Step2: Data Preprocessing-1	23-Feb	8-Mar					
Step3: Data Preprocessing-2	23-Feb	8-Mar					
Step4: LSA	23-Feb	8-Mar					
Step5: Logistic Regression	5-Mar	15-Mar					
Step6: Machine Learning	5-Mar	15-Mar					
Key Takeaways							
Main Findings	5-Mar	15-Mar					
Blog Post	8-Mar						
Moving on							

Progress Check

FOMC Minutes Analysis



Web Scrapping	**100%: Complete Scraping of FOMC data**
Data Pre-processing 1	**100%: pdf2txt, stop word removal, tokenization, bigram, lemmatization, etc**
Data Pre-processing 2	**100%: bag of words, tf-idf, scaling the data, import interest rate data, etc.**
LSA	**100%: Dimensionality reduction, calculate document similarity**
Logistic regression	**100%: Linear model for testing the individual effect of each of many regressors**
Machine Learning	**100%: Research findings and Machine learning for prediction**

4D-Intelli

Main Findings

Take-aways

1. It's hard to simply use FOMC minutes to predict the actual change of the Fed Rate, and every statistic model performs poorly for that.

2. There are some prediction power in terms of the sign of the Fed Rate change (up or down), even conditioning on the internal information.

3. The prediction power is even more significant when average the interest rate change by year, although the data would be much small and therefore might be less reliable.

4. We can get a better prediction by focusing on some industry performance such as the real estate industry on which the interest rate plays an import role.

Moving On…

Things to improve

1 **Data Preprocessing**

- There is some mistakes in our data preprocessing, some unexpected words may come up

2 **IR and Model Selection**

- IR Data Selection: Annual average IR data v.s. 8 times per year
- Model Selection: NLP – Bow + tfidf, ML – some model

3 **Industry Data**

- Now we apply our findings to REIT interest rate to focus on real estate industry, and we can try other industries in the future.