People's Daily Microblog Simulator

– Application of Machine Learning Models to Assist Editors to Evaluate Post Performance –

Anran Wang, Tian Jin, Zhenming Wang

Table of contents

01. Introduction

Research Question: Given a piece of text from *People's Daily* Microblog, how would choices of words influence the popularity of the post (forwards, likes, comments)?

Expectation: We propose to build a *People's Daily* Microblog Simulator, where users can type in the text they'd like to post as *People's Daily*, and our simulator will automatically evaluate the performance of the post by predicting the numbers of likes, comments, and forwards. Benefitting from such simulator, true *People's Daily* editors could accordingly revise their posts before sending them out. To the most promising, the mechanism could be further applied to any other bloggers.

人民日报
PEOPLE'S DAILY

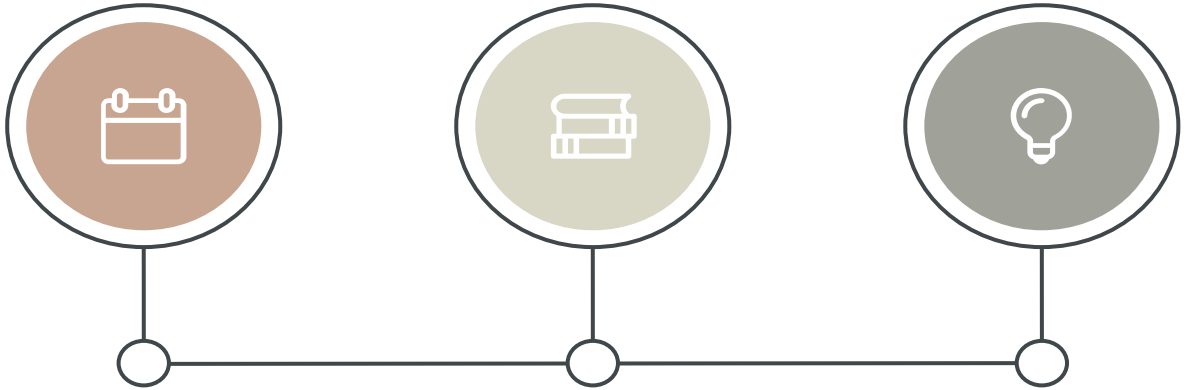
02. Dataset: Weibo Spider

https://github.com/dataabc/weiboSpider

P13 fx

	A	B	C	D	E	F	G	H	I	J	K	L
1	微博id	微博正文	头条文章url	原始图片url	微博视频url	发布位置	发布时间	发布工具	点赞数	转发数	评论数	
2	KdsPA3mAh	【五一当天，#云南多地		http://ww2.s	无	无	2021/5/1 17:05	微博 weibo.	989	90	198	
3	KdsHxxrLD	【惊艳！#杭州小伙用千		无	http://f.vide	无	2021/5/1 16:46	微博视频号	1606	280	173	
4	Kdsw72II7	【#迎客松守松人一年驻		无	http://locall	无	2021/5/1 16:18	微博视频号	1191	128	160	
5	KdsgR9lxH	【这段视频告诉你，#张		无	http://f.vide	无	2021/5/1 15:41	微博视频号	3526	367	289	
6	Kds0ztD7q	【#农民工怕弄脏座位坐		无	http://f.vide	无	2021/5/1 15:01	微博视频号	45528	1774	1542	
7	KdrV7r58n	【北京铁路：#京广高铁		http://wx3.s	无	无	2021/5/1 14:47	微博 weibo.	5448	332	858	
8	KdrMPch0h	【#云南将现1.5亿只蝴蝶		无	http://f.vide	无	2021/5/1 14:27	微博视频号	51993	2192	2305	
9	KdrDF0HKc	【避开人山人海！#五一		无	无	无	2021/5/1 14:04	微博原生直	3100	125	341	
10	Kdrm0bCJy	【#遛狗不栓绳危害有多		无	http://locall	无	2021/5/1 13:21	微博视频号	6655	417	1086	
11	Kdrcvl3Fh	【#5400余名北大师生5.4		无	http://f.vide	无	2021/5/1 12:57	微博视频号	2906	248	335	
12	KdqXvEvK7	【#黑猩猩酷爱洗衣服被		无	http://f.vide	无	2021/5/1 12:20	微博视频号	11707	1061	1228	
13	KdqJYba35	【#劳动节版你笑起来真		无	无	无	2021/5/1 11:47	微博视频号	5885	957	502	
14	Kdqwa6xwH	【周知！#多部门发布五		http://ww1.s	无	无	2021/5/1 11:13	微博 weibo.	19178	557	528	
15	Kdqeq0G2h	【#江苏南通大风致11死		无	http://f.vide	无	2021/5/1 10:29	微博视频号	98816	5676	5686	
16	Kdq6O8MBy	【正在直播：#与一线劳		无	http://live.v	无	2021/5/1 10:10	微博原生直	6824	198	440	
17	Kdq3t0rBv	【#王进喜珍贵原声首次		无	http://f.vide	无	2021/5/1 10:02	微博视频号	5185	567	604	
18	KdpTV4QUy	【搜狗输入法、高德地图		http://ww2.s	无	无	2021/5/1 09:39	微博 weibo.	10561	652	876	
19	KdpMz8bnD	【#假期去海边不要食用		http://ww4.s	无	无	2021/5/1 09:21	微博 weibo.	35066	2512	2783	
20	KdptbENLm	【#百年劳动者面孔混剪		无	无	无	2021/5/1 08:33	微博视频号	5821	2230	1290	
21	Kdpk4prl7	【#人民日报五一社论#		无	无	无	2021/5/1 08:10	微博 weibo.	2446	609	355	
22	Kdp9xqw4r	【仅剩67位！#南京大屠		http://wx2.s	无	无	2021/5/1 07:44	微博 weibo.	11247	654	1559	
23	Kdp1Z3aS9	【今天，五一劳动节。发		http://wx2.s	无	无	2021/5/1 07:26	微博 weibo.	9262	3515	1503	
24	KdoZqCMe4	【#美国宣布限制印度旅		http://wx1.s	无	无	2021/5/1 07:20	微博 weibo.	44124	703	1744	
25	KdoRpzwQ6	【你好，#五月#】5月，		http://ww3.s	无	无	2021/5/1 07:00	微博 weibo.	6771	3074	1523	
26	KdlZdsFGQ	【#你好，明天#】多部门		http://wx3.s	无	无	2021/4/30 23:41	微博 weibo.	177148	868	2533	

03. Methodology



Pre-processing

Jiaba
Data Normalization
~~Bag of Words~~
TF-IDF (Sparse Matrix)

Models

Linear Regression
Random Forest
SVM
Neural Network

Evaluation

MAE

03/ Pre-processing: Regex + Jieba

Before:

【#黑猩猩酷爱洗衣服被评劳模#，网友：比我男朋友还勤快】4月30日，重庆一主题公园，一只黑猩猩因酷爱洗衣服被评为“动物界劳模”。这只名叫渝辉的黑猩猩每次看到饲养员洗衣服都十分兴奋，还爱看书看报，被网友称为“暖男”。公园为它颁发了定制版荣誉证书，还准备了丰厚的美食。时间视频的微博视频

【#劳动节版你笑起来真好看#】生活中，哪一种笑最打动人？今天，向劳动者致敬！#让劳动者上热搜# 人民日报的微博视频

【周知！#多部门发布五一假期提示#】①#五一期间仍可接种疫苗#，全国疫苗接种工作“不打烊”；②乘坐公共交通全程佩戴口罩；③“五一”假期景区全面实施门票预约制；④提倡错峰出行，提前做好行程规划；⑤如旅行中出现健康异常，就地诊治…#五一假期的9个健康提示#↓↓转给你关心的TA！ [组图共9张] 原图

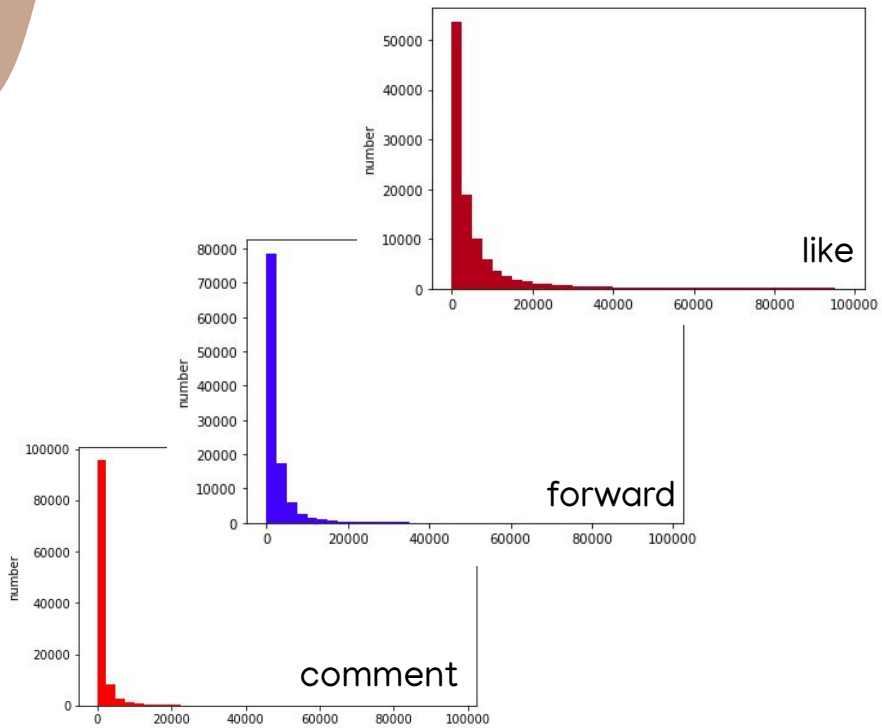
【#江苏南通大风致11死102伤#】据@南通发布，4月30日18时至22时，江苏南通部分地区出现冰雹和大范围强雷暴大风天气。截至目前，受灾人口3000余人，其中因大树倒伏砸倒房屋、电线杆刮断、狂风卷入河道等原因，造成11人死亡，因灾受伤人口102人。苏海门渔01728倾覆，11名船员落水，2人已成功获救，9人仍在紧张搜救中。江苏新闻的微博视频

【正在直播：#与一线劳动者一起现场作业#】5月1日，国际劳动节，@人民日报 联合天目新闻，多路联动，与铁路维修工、黄山风景区挑夫、公路养护人员一起现场作业，向不同工种的一线劳动者致敬！戳直播↓↓#让劳动者上热搜#！ 人民日报的微博视频

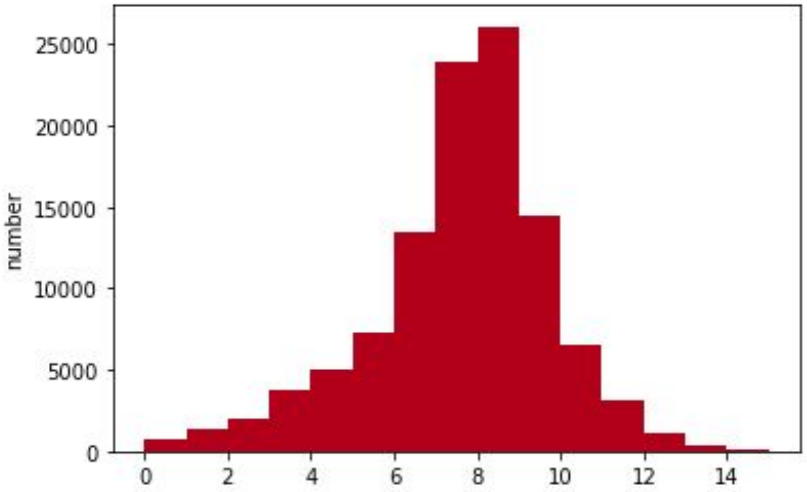
After:

```
time: 25.127901315689087
['黑猩猩', '酷爱', '洗衣服', '评', '劳模', '网友', '男朋友', '勤快', '4月30日',
'重庆', '主题公园', '一只', '黑猩猩', '酷爱', '洗衣服', '评为', '动物界', '劳
模', '名叫', '渝辉', '黑猩猩', '每次', '看到', '饲养员', '洗衣服', '十分', '兴
奋', '爱', '看书', '看报', '网友', '称为', '暖男', '公园', '颁发', '定制',
'版', '荣誉', '证书', '准备', '丰厚', '美食', '时间', '视频', '微博', '视频']
['劳动节', '版', '笑起来', '真好看', '生活', '中', '一种', '笑', '打动', '今
天', '劳动者', '致敬', '劳动者', '热', '搜', '人民日报', '微博', '视频']
['周知多', '部门', '发布', '五一', '假期', '提示', '①', '五一', '期间', '仍
可', '接种', '疫苗', '全国', '疫苗', '接种', '工作', '打烊', '②', '乘坐', '公
共交通', '全程', '佩戴', '口罩', '③', '五一', '假期', '景区', '全面', '实施',
'门票', '预约制', '④', '提倡', '错峰', '出行', '提前', '做好', '行程', '规划',
'⑤', '旅行', '中', '出现', '健康', '异常', '就地', '诊治', '五一', '假期', '9
个', '健康', '提示', '转给', '关心', 'TA', '组图', '共', '9张', '原图']
['江苏', '南通', '大风', '11', '死', '102', '伤', '南通', '发布', '4月30日18时
至22时', '江苏', '南通', '部分', '地区', '出现', '冰雹', '范围', '强', '雷暴',
'大风', '天气', '目前', '受灾', '人口', '3000余', '大树', '倒伏', '砸', '倒',
'房屋', '电线杆', '刮断', '狂风', '卷入', '河道', '原因', '造成', '11', '死亡',
'灾', '受伤', '人口', '102', '苏海门渔01728', '倾覆', '11名', '船员', '落水',
'成功', '获救', '紧张', '搜救', '中', '江苏', '新闻', '微博', '视频']
['正在', '直播', '一线', '劳动者', '一起', '现场', '作业', '5月1日', '国际劳动节
人民日报联合天目', '新闻', '多路', '联动', '铁路', '维修工', '黄山风景区', '挑夫公
路', '养护', '人员', '一起', '现场', '作业', '不同', '工种', '一线', '劳动者',
'致敬', '戳', '直播', '劳动者', '热', '搜', '人民日报', '微博', '视频']
```

03/ Pre-processing: Data Normalization

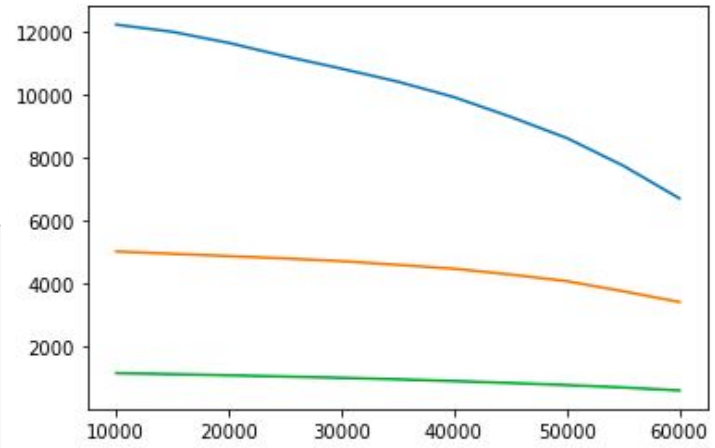


Log transformation



03/ Pre-processing: TF-IDF

- Term frequency and inverse document frequency
- Hyperparameter: max_features




```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer


vectorizer = CountVectorizer(max_features = 10000)

tf_idf_transformer = TfidfTransformer()

tf_idf1 = tf_idf_transformer.fit_transform(vectorizer.fit_transform(X_train))
X_train_weight = tf_idf1

tf_idf2 = tf_idf_transformer.transform(vectorizer.transform(X_test))
X_test_weight = tf_idf2
```

03/ Pre-processing: Sparse Matrix

```
train_data_set = MyDataSet(train_x, train_y)
train_loader = DataLoader(dataset=train_data_set, batch_size=batchSize, shuffle=Tr

# test_data:
X_test_weight_nn = X_test_weight_nn.astype(float)
y_test_nn = y_test_nn.astype(float)
#test_x = convert_sparse_matrix_to_sparse_tensor(X_test_weight_nn)
test_x = torch.from_numpy(X_test_weight_nn)
test_y = torch.from_numpy(y_test_nn)

test_data_set = MyDataSet(test_x, test_y)
test_loader = DataLoader(dataset=test_data_set, batch_size=batchSize, shuffle=True
```

Your session crashed after using all available RAM. View runtime logs

55s completed at 12:30 AM

```
  (0, 53970)    0.15875208796311152
  (0, 53636)    0.29413458769821654
  (0, 51813)    0.12432949999034688
  (0, 51169)    0.1783624739121948
  (0, 50346)    0.14116579425427336
  (0, 48027)    0.14114401037214977
  (0, 47645)    0.25078925455808043
  (0, 45608)    0.16023239575168496
  (0, 43495)    0.1652704605370866
  (0, 42920)    0.13590345321628972
  (0, 40694)    0.21752005098874752
  (0, 37400)    0.14897036813493925
  (0, 35464)    0.14084082555747807
  (0, 33895)    0.255827319343482
  (0, 29268)    0.16954746823947583
  (0, 24109)    0.13951621309098325
  :     :
(33074, 61474)        0.14671055721583312
(33074, 61453)        0.11350125474984606
(33074, 58310)        0.3597110268688744
(33074, 58077)        0.13218140037257267
(33074, 47229)        0.17170867343477023
(33074, 47086)        0.09849741146770362
(33074, 45585)        0.2657150952377177
(33074, 42955)        0.164011259153423
(33074, 40761)        0.09668255080219637
(33074, 37200)        0.13945608115326913
(33074, 34250)        0.2670403876216512
(33074, 33852)        0.13282490302713665
(33074, 31649)        0.1314318618599812
(33074, 26136)        0.1680540259577002
(33074, 23467)        0.17583596236027654
(33074, 20907)        0.09567491617082742
(33074, 20469)        0.0314662757929653
(33074, 19954)        0.24110037645779683
```

Toarray -> sparse matrix

03/ Model: Linear Regression

```
train_error = sklearn.metrics.mean_absolute_error(np.exp(y_train.to_numpy()), np.exp(y_train_pred),
                                                  multioutput='raw_values')
print(train_error)

test_error = sklearn.metrics.mean_absolute_error(np.exp(y_train.to_numpy()), np.exp(y_train_pred),
                                                 multioutput='raw_values')
print(test_error)
```

```
[12240.8485751   5037.8056129   1173.41195107]
[13317.31412094  5205.49306221  1221.45838774]
```

```
new = np.array(['【宁波24岁姑娘徒步西藏墨脱失踪9天】小许，女，24岁，宁波慈溪人，身高160cm，戴眼镜。7月14日，小许带着2000元独自出发
                骑行318川藏线，到八一后准备徒步去墨脱。8月17日晚，她跟两个男生逃票进入派乡，之后失去联系！目前，当地警方已经介入开始
                找人。祝平安！慈溪24岁姑娘徒步西藏墨脱失踪9天了还没有消息·都市快报 @都市快报   原图 '])
weight = tf_idf_transformer.transform(vectorizer.transform(new))
pred = model.predict(weight)
print(np.exp(pred))
```

```
[[191.11886701 871.25871791 826.90838998]]
```

```
true: 125  1332  886
```

- Linear model doesn't perform well; more complex, non-linear model needed
- Multi-output regression? correlation?

03/ Model: Support Vector Regression

```
svr = sklearn.svm.SVR(kernel ='rbf',degree = 3, gamma ='auto', coef0 = 0.0, tol = 0.001, C = 1.0,
epsilon = 0.1, shrinking = True, cache_size = 200, verbose = False, max_iter = -1)

svr.fit(X_train_weight, y_train['like'])
y_train_pred0 = svr.predict(X_train_weight)
y_test_pred0 = svr.predict(X_test_weight)

svr.fit(X_train_weight, y_train['forward'])
y_train_pred1 = svr.predict(X_train_weight)
y_test_pred1 = svr.predict(X_test_weight)

svr.fit(X_train_weight, y_train['comment'])
y_train_pred2 = svr.predict(X_train_weight)
y_test_pred2 = svr.predict(X_test_weight)
```

3 models for 3-output regression

Hyperparameter: kernel (rbf / poly / sigmoid); degree

03/ Model: Support Vector Regression

```
rbf


like       5002.900623
forward    4422.304667
comment    1188.533845
dtype: float64
like       4909.050937
forward    3373.718870
comment    1213.058375
dtype: float64


poly

like       5618.077962
forward    4660.494981
comment    1190.669840
dtype: float64
like       5512.656311
forward    3612.471748
comment    1215.079982
dtype: float64
```

- The best performanced model
- Might solve the problem of correlation
- Degree not tuned

03/ Model: Random Forests

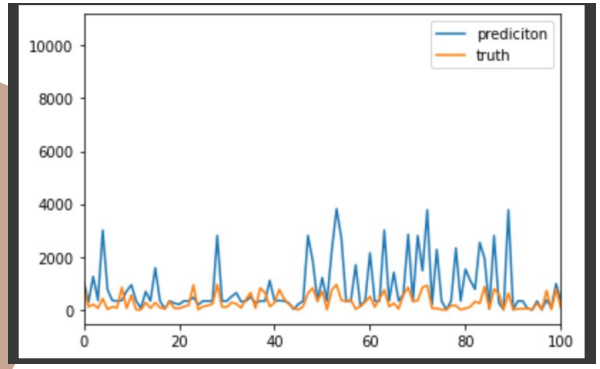
Number of trees: 150

Trained with different settings:

- max_features = 10000
- max_features = 100000
- depth of trees = 10
- depth of trees = 20

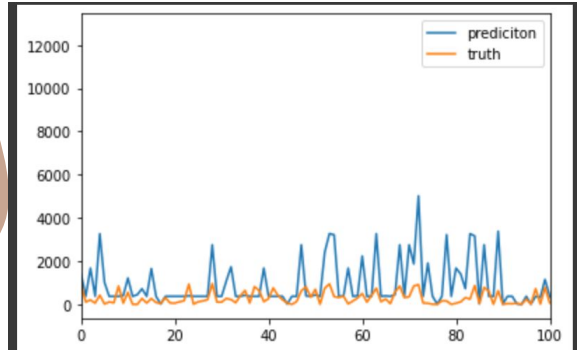
```
model = RandomForestRegressor(n_estimators=150,max_depth=10, random_state=0)

model.fit(X_train,y_train)

print("training done")
```

03/ Model: Random Forests



Max_feature = 10000, tree_depth = 20

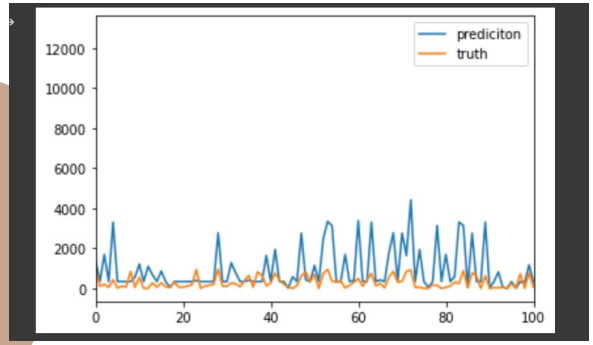
```
MAE ON TRAINING SET BEFORE TAKING EXPONENTIAL:  [1.12969505 0.73334146 0.76983594]
MAE ON TRAINING SET AFTER TAKING EXPONENTIAL:  [11009.02247832  4933.05269566  1168.42722616]
MAE ON TESTING SET BEFORE TAKING EXPONENTIAL:  [1.23093057 0.80077331 0.8506217 ]
MAE ON TESTING SET AFTER TAKING EXPONENTIAL:  [11707.99202964  5201.86282374  1196.33314958]
```



Max_feature = 10000, tree_depth = 10

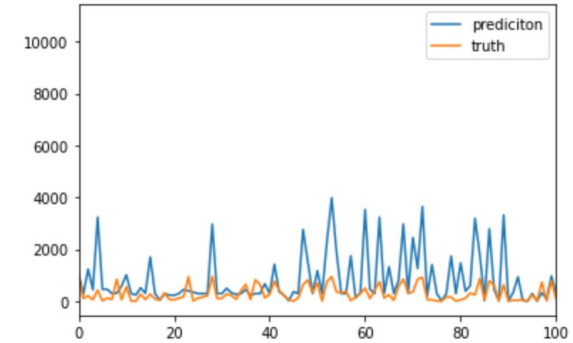
```
MAE ON TRAINING SET BEFORE TAKING EXPONENTIAL:  [1.28785163 0.82116102 0.86749969]
MAE ON TRAINING SET AFTER TAKING EXPONENTIAL:  [11947.64261715  5211.44817453  1258.9984395 ]
MAE ON TESTING SET BEFORE TAKING EXPONENTIAL:  [1.31265706 0.83793342 0.89070597]
MAE ON TESTING SET AFTER TAKING EXPONENTIAL:  [11893.85688217  5255.43325532  1219.87169656]
```

03/ Model: Random Forests



Max_feature = 100000, tree_depth = 10

```
MAE ON TRAINING SET BEFORE TAKING EXPONENTIAL:  [1.27939306 0.81600587 0.86379184]
MAE ON TRAINING SET AFTER TAKING EXPONENTIAL:  [11943.25924728  5199.47662184  1257.24292804]
MAE ON TESTING SET BEFORE TAKING EXPONENTIAL:  [1.30820185 0.83518134 0.88865596]
MAE ON TESTING SET AFTER TAKING EXPONENTIAL:  [11894.3467885   5250.0399778   1218.54709298]
```



Max_feature = 100000, tree_depth = 20

```
MAE ON TRAINING SET BEFORE TAKING EXPONENTIAL:  [1.12319289 0.72993204 0.7665977 ]
MAE ON TRAINING SET AFTER TAKING EXPONENTIAL:  [11004.5874486   4918.86775181  1165.77679719]
MAE ON TESTING SET BEFORE TAKING EXPONENTIAL:  [1.22716085 0.79956614 0.85020987]
MAE ON TESTING SET AFTER TAKING EXPONENTIAL:  [11702.07836839  5199.42832755  1195.86436983]
```

03/ Model: Neural Network

Neural Network Structure

```
import torch
from torch.autograd import Variable
import torch.nn.functional as F
import matplotlib.pyplot as plt

class Net(torch.nn.Module):
  def __init__(self, n_feature, n_hidden, n_output):
    super(Net, self).__init__()
    self.hidden = torch.nn.Linear(n_feature, n_hidden)
    self.predict = torch.nn.Linear(n_hidden, n_output)

  def forward(self, x):
    x = F.relu(self.hidden(x))
    x = self.predict(x)
    return x

net = Net(10000, 10, 3)
print(net)
net = net.double()

optimizer = torch.optim.SGD(net.parameters(), lr=0.5)
loss_function = torch.nn.L1Loss()

plt.ion()
plt.show()
```

```
Net(
  (hidden): Linear(in_features=10000, out_features=10, bias=True)
  (predict): Linear(in_features=10, out_features=3, bias=True)
)
```

from torch.utils.data.dataloader import DataLoader
from torch.utils.data.dataset import Dataset

```
batchSize = 256
epochNum = 10
# inputSlice = 20000
```

```
from torch.utils.data.dataloader import DataLoader
from torch.utils.data.dataset import Dataset
import torch.utils.data as Data
import torch

class MyDataSet():
    def __init__(self, inputX, inputY):
        super().__init__()
        self.X = inputX
        self.Y = inputY
        self._len = len(self.X)
        pass
    def __getitem__(self, idx):
        return (self.X[idx], self.Y[idx])

    def __len__(self):
        return self._len

train_data_set = MyDataSet(train_x, train_y)
train_loader = DataLoader(dataset=train_data_set, batch_size=batchSize, shuffle=True)

test_data_set = MyDataSet(test_x, test_y)
test_loader = DataLoader(dataset=test_data_set, batch_size=batchSize, shuffle=True)
```

```
# neuron:5
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12737.46688887  1334.04577996  5574.20811499]
TEST ERROR!!
[12573.91204917  1375.28546174  9504.5288857 ]
```

```
# neuron:7
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12631.66932127  1349.04199398  5548.54195958]
TEST ERROR!!
[12509.48492121  1458.41987516  9586.88086884]
```

```
# neuron:10
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12527.43855916  1254.24672193  5437.73017198]
TEST ERROR!!
[12307.77097221  1302.55125891  9360.99570437]
```

```
# neuron:15
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12553.71922625  1262.64620268  5450.06649833]
TEST ERROR!!
[12461.96738681  1339.15609752  9433.16608345]
```

```
# neuron:20
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12662.9252431   1352.55885474  5574.63375898]
TEST ERROR!!
[12505.0915534   1414.75197035  9553.57151403]
```

Tune #Neuron: 10

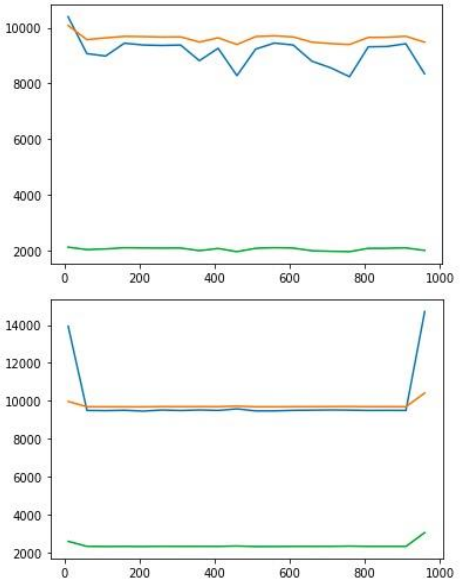
```
# neuron:10
batchSize: 512 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12568.35569375  1289.12045663  5485.02483937]
TEST ERROR!!
[12435.42117523  1338.55812754  9433.197557  ]
```

```
# neuron:10
batchSize: 256 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12527.43855916  1254.24672193  5437.73017198]
TEST ERROR!!
[12307.77097221  1302.55125891  9360.99570437]
```

```
# neuron:10
batchSize: 128 epochNum: 1 inputSlice: 10000
TRAIN ERROR!!
[12456.39723128  1246.77121704  5418.0303683 ]
TEST ERROR!!
[12369.86210662  1320.58609568  9394.37296849]
```

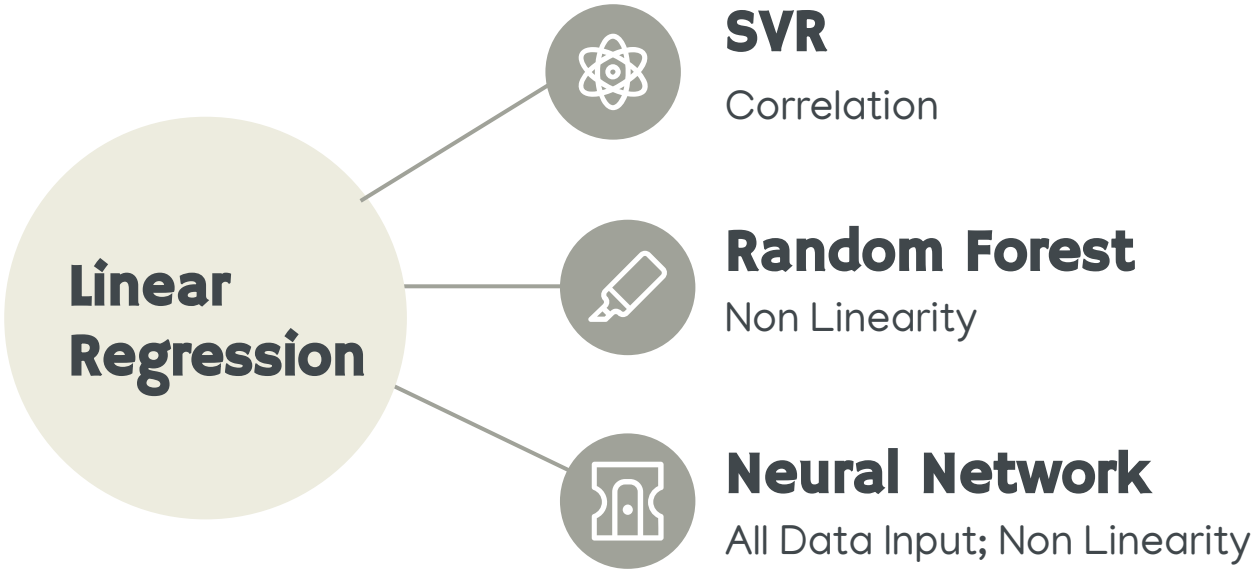
Tune #batchSize: 256

Epoch Vs. Like_error



Dataset=5000

04. Result: Model Comparison



Linear Regression: # Input: all

```
train_error = np.absolute(np.subtract(np.exp(y_train.to_numpy()),np.exp(y_train_pred))).mean(axis=0)
print(train_error)

test_error = np.absolute(np.subtract(np.exp(y_test.to_numpy()),np.exp(y_test_pred))).mean(axis=0)
print(test_error)
```

```
[12240.8485751   5037.8056129   1173.41195107]
[13317.31412094  5205.49306221  1221.45838774]
```

One-layer NN

Input: all
BatchSize = 256
Neurons = 10
Epoch = 20

```
TRAIN ERROR!!
[11670.063841632777, 5146.812287880117, 1211.3048044133116]
TEST ERROR!!
[11446.500611296187, 5098.407987872186, 1116.831272920347]
```

Random Forest

Input: all
Tree = 150
Depth = 20

```
MAE ON TRAINING SET BEFORE TAKING EXPONENTIAL:
[1.12319289 0.72993204 0.7665977 ]
MAE ON TRAINING SET AFTER TAKING EXPONENTIAL:
[11004.5874486   4918.86775181  1165.77679719]
MAE ON TESTING SET BEFORE TAKING EXPONENTIAL:
[1.22716085 0.79956614 0.85020987]
MAE ON TESTING SET AFTER TAKING EXPONENTIAL:
[11702.07836839  5199.42832755  1195.86436983]
```

```
like        5002.900623
forward     4422.304667
comment     1188.533845
dtype: float64
like        4909.050937
forward     3373.718870
comment     1213.058375
dtype: float64
```

```
like        5002.900623
forward     4422.304667
comment     1188.533845
dtype: float64
like        4909.050937
forward     3373.718870
comment     1213.058375
dtype: float64

50000data
```

SVR

# Input = 50000 / all
Kernel = “rbf”
Degree = 3

04. Result: Weibo Interface

Interface for posting as *People's Daily* on Weibo

+ Integrate a model on server

05. Reflections

Challenges

- Data crawling
- SUPER large dataset causing SUPER long training time
- High dimensionality
- Time Limit
- AB Power Failure

Future Improvement

- **Pre-processing:** number of Max_feature
- **Model:** multiple-layer neural network
- **Tuning:** using full dataset
- Sentiment Analysis of Comments

06/ Reference

- **Weibo Spider**: https://github.com/dataabc/weiboSpider
- **Jieba**: https://github.com/fxsjy/jieba

/ Special Thanks to:

- **Prof. Guo**
- **Github**
- **CSDN**

Thank you!

2021 Spring Machine Learning