Week13-1

November 25, 2024

1 Week 13-1 - RANDOM FORESTS AND BOOSTING

This lecture is comprised of 4 parts:

- 1. CART Baseline
- 2. Random Forests (basic model and cross-validation)
- 3. Gradient Boosted Trees (basic model and cross-validation)
- 4. Final Comparison

[41]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

[42]:
```python
ctr = pd.read_csv("CTR.csv")
ctr.info()
ctr.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6057 entries, 0 to 6056
Data columns (total 11 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   CTR            6057 non-null   float64
 1   titleWords     6057 non-null   int64
 2   adWords        6057 non-null   int64
 3   depth          6057 non-null   int64
 4   position       6057 non-null   int64
 5   advCTR         6057 non-null   float64
 6   advCTRInPos    6057 non-null   float64
 7   queryCTR       6057 non-null   float64
 8   queryCTRInPos  6057 non-null   float64
 9   gender         6057 non-null   object
 10  age            6057 non-null   object
dtypes: float64(5), int64(4), object(2)
memory usage: 520.7+ KB
```

[42]:
```
      CTR  titleWords  adWords  depth  position  advCTR  advCTRInPos  \
0  0.0000           8       17      1         1  0.0136       0.0153
1  0.0000           9       19      3         3  0.0199       0.0088
```

```
2  0.0675             6        30       2           1  0.0825        0.1002
3  0.0000             5        19       3           2  0.0116        0.0090
4  0.0000            10        22       1           1  0.0186        0.0284

   queryCTR  queryCTRInPos  gender    age
0    0.0000         0.0000    male   0-12
1    0.0394         0.0125    male  25-30
2    0.0200         0.0256  female  13-18
3    0.0042         0.0017  female  25-30
4    0.0294         0.0431  female   0-12
```

[43]:
```python
def OSR2(model, X_test, y_test, y_train):

    y_pred = model.predict(X_test)
    SSE = np.sum((y_test - y_pred)**2)
    SST = np.sum((y_test - np.mean(y_train))**2)

    return (1 - SSE/SST)
```

1.1 1. CART BASELINE

1.1.1 Decision Tree Regressor with CV

We will use a standard decision tree to establish a reference against which to evaluate the ensemble models

1.0 Train test split

[44]:
```python
from sklearn.model_selection import train_test_split

y = ctr['CTR']
X = pd.get_dummies(ctr.drop(['CTR'], axis=1))

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=88)
X_train.shape, X_test.shape
```

[44]: ((4239, 18), (1818, 18))

1.1 Define the grid values and perform the Grid Search Cross-Validation

[61]:
```python
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import KFold

grid_values = {'ccp_alpha': np.linspace(0, 0.001, 51)}

dtr = DecisionTreeRegressor(min_samples_leaf=5, min_samples_split=20,␣
 ↪random_state=88)
```

```
### Note that the line below is important. It ensures that the training data is␣
↪split into
### five folds randomly. Recall what we've seen in the discussion slides that␣
↪by default,
### GridSearchCV will split the training data without shuffling.
cv = KFold(n_splits=5,random_state=1,shuffle=True)
### by setting random_state as a fixed number, we ensure that each time the␣
↪GridSearchCV splits data, we get the
### same split.
dtr_cv = GridSearchCV(dtr, param_grid=grid_values, scoring='r2', cv=cv,␣
↪verbose=0)
dtr_cv.fit(X_train, y_train)
```

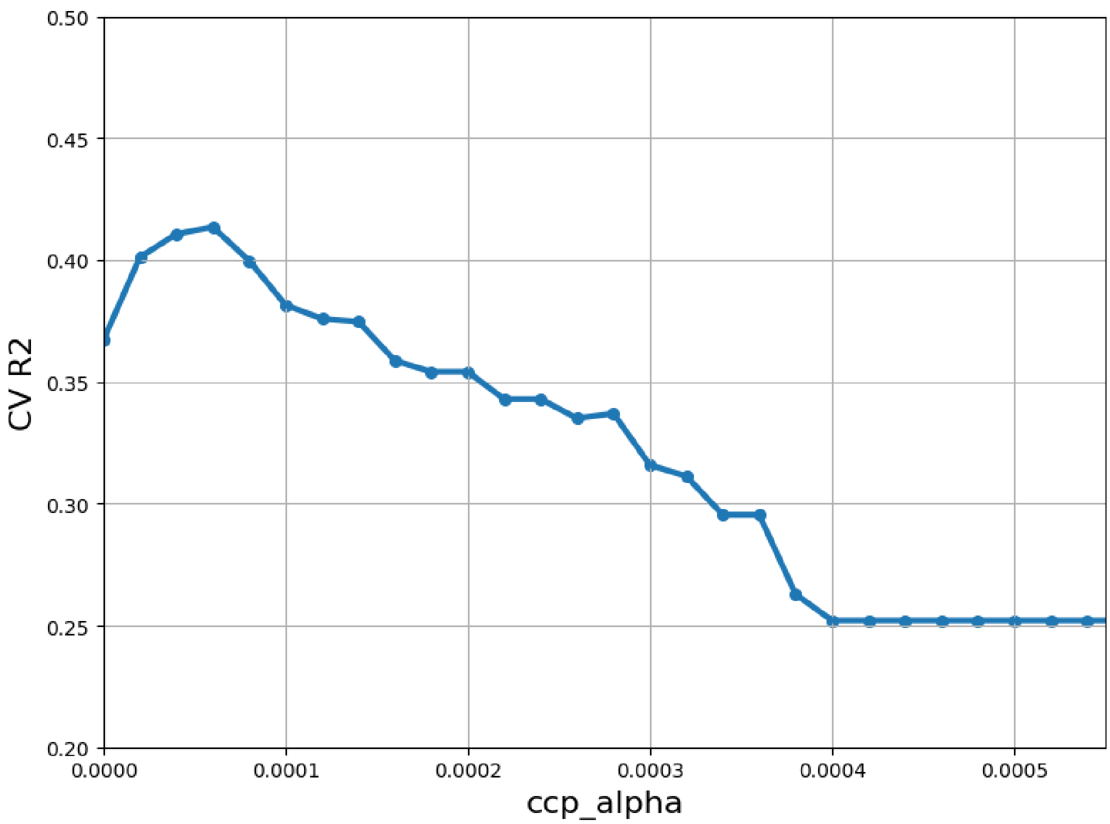
[61]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             estimator=DecisionTreeRegressor(min_samples_leaf=5,
                                             min_samples_split=20,
                                             random_state=88),
             param_grid={'ccp_alpha': array([0.0e+00, 2.0e-05, 4.0e-05, 6.0e-05,
8.0e-05, 1.0e-04, 1.2e-04,
       1.4e-04, 1.6e-04, 1.8e-04, 2.0e-04, 2.2e-04, 2.4e-04, 2.6e-04,
       2.8e-04, 3.0e-04, 3.2e-04, 3.4e-04, 3.6e-04, 3.8e-04, 4.0e-04,
       4.2e-04, 4.4e-04, 4.6e-04, 4.8e-04, 5.0e-04, 5.2e-04, 5.4e-04,
       5.6e-04, 5.8e-04, 6.0e-04, 6.2e-04, 6.4e-04, 6.6e-04, 6.8e-04,
       7.0e-04, 7.2e-04, 7.4e-04, 7.6e-04, 7.8e-04, 8.0e-04, 8.2e-04,
       8.4e-04, 8.6e-04, 8.8e-04, 9.0e-04, 9.2e-04, 9.4e-04, 9.6e-04,
       9.8e-04, 1.0e-03])},
             scoring='r2')
```

1.2 Select the best hyperparameter

[7]:
```
ccp_alpha = dtr_cv.cv_results_['param_ccp_alpha'].data
R2_scores = dtr_cv.cv_results_['mean_test_score']

plt.figure(figsize=(8, 6))
plt.xlabel('ccp_alpha', fontsize=16)
plt.ylabel('CV R2', fontsize=16)
plt.scatter(ccp_alpha, R2_scores, s=30)
plt.plot(ccp_alpha, R2_scores, linewidth=3)
plt.grid(True, which='both')
plt.xlim([0, 0.00055])
plt.ylim([0.2, 0.5])

plt.tight_layout()
plt.show()
```



[8]:
```python
print('Best ccp_alpha', dtr_cv.best_params_)
```

```
Best ccp_alpha {'ccp_alpha': 6.000000000000001e-05}
```

1.3 Evaluate the model performance (trained on the entire training set)

[9]:
```python
# Model Evaluation
print('Cross-validated R2:', round(dtr_cv.best_score_, 5))
print('OSR2:', round(OSR2(dtr_cv, X_test, y_test, y_train), 5))
```

```
Cross-validated R2: 0.41349
OSR2: 0.4819
```

1.2 2. RANDOM FORESTS

1.2.1 2.1 Random Forest Regressor

[10]:
```python
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(max_features=5, min_samples_leaf=5,
                           n_estimators = 500, random_state=88, verbose=1)
# Note: you can change the verbose parameter to control how much training␣
↪progress is printed.
```

```
rf.fit(X_train, y_train)
```

```
[Parallel(n_jobs=1)]: Done  49 tasks      | elapsed:    0.2s
[Parallel(n_jobs=1)]: Done 199 tasks      | elapsed:    0.9s
[Parallel(n_jobs=1)]: Done 449 tasks      | elapsed:    2.0s
```

[10]:
```
RandomForestRegressor(max_features=5, min_samples_leaf=5, n_estimators=500,
                      random_state=88, verbose=1)
```

[11]:
```
rf.verbose = False

print('OSR2:', round(OSR2(rf, X_test, y_test, y_train), 5))
```

```
OSR2: 0.56467
```

Feature Importance

[12]:
```
pd.DataFrame({'Feature' : X_train.columns,
              'Importance score': 100*rf.feature_importances_}).round(1)
```

[12]:

	Feature	Importance score
0	titleWords	3.7
1	adWords	3.4
2	depth	2.3
3	position	2.2
4	advCTR	23.6
5	advCTRInPos	32.6
6	queryCTR	10.0
7	queryCTRInPos	17.9
8	gender_female	0.8
9	gender_male	0.5
10	gender_unknown	0.8
11	age_0-12	0.0
12	age_13-18	0.3
13	age_19-24	0.5
14	age_25-30	0.3
15	age_31-40	0.2
16	age_41+	0.0
17	age_unknown	0.8

1.2.2 2.2 Random Forest Regressor with CV

2.2.1 Define the grid values and perform the Grid Search Cross-Validation

[13]:
```
import time

grid_values = {'max_features': np.linspace(1,25,25, dtype='int32'),
               'min_samples_leaf': [5],
               'n_estimators': [40],
               'random_state': [88]}
```

```
tic = time.time()

rf2 = RandomForestRegressor()
# Note: here we set verbose=2 to keep track of the progress (the running time)␣
↪of the cross validation.
cv = KFold(n_splits=5,random_state=2333,shuffle=True)
rf_cv = GridSearchCV(rf2, param_grid=grid_values, scoring='r2', cv=cv,verbose=1)
rf_cv.fit(X_train, y_train)

toc = time.time()

print('time:', round(toc-tic, 2),'s')
```

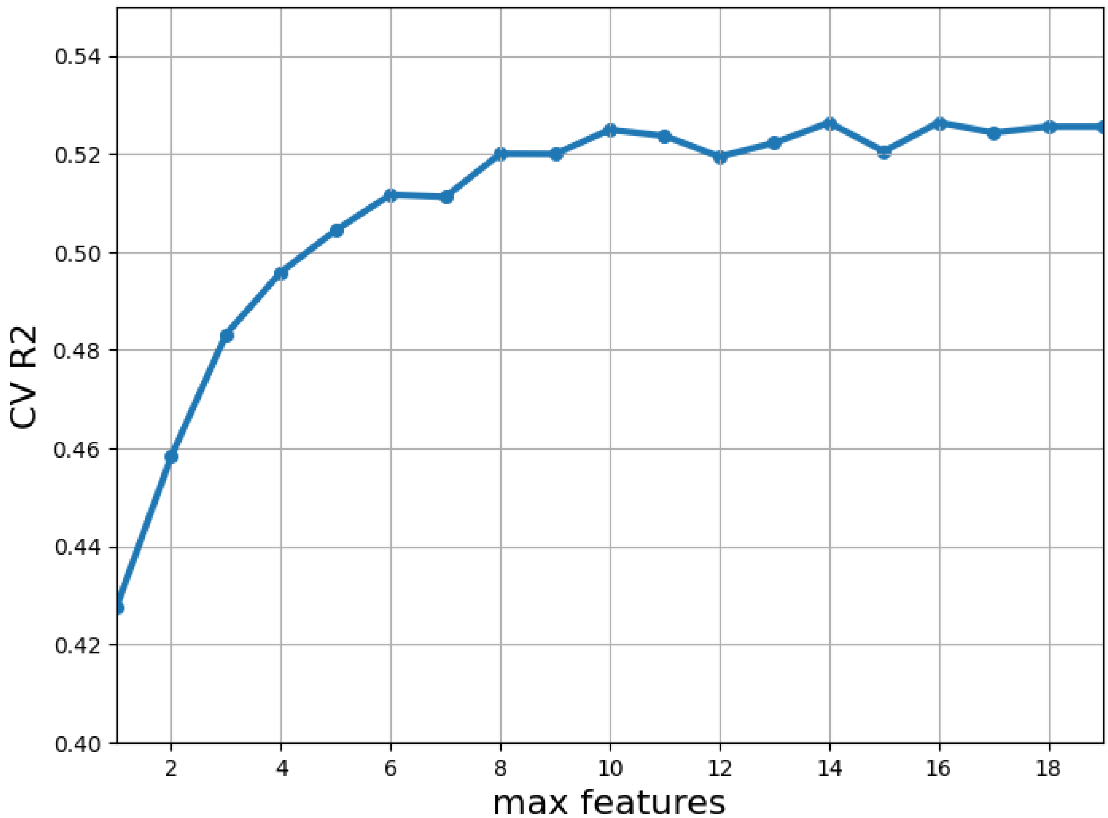
```
Fitting 5 folds for each of 25 candidates, totalling 125 fits
time: 37.07 s
```

2.2.2 Select the best hyperparameter

[14]:
```
max_features = rf_cv.cv_results_['param_max_features'].data
R2_scores = rf_cv.cv_results_['mean_test_score']

plt.figure(figsize=(8, 6))
plt.xlabel('max features', fontsize=16)
plt.ylabel('CV R2', fontsize=16)
plt.scatter(max_features, R2_scores, s=30)
plt.plot(max_features, R2_scores, linewidth=3)
plt.grid(True, which='both')
plt.xlim([1, 19])
plt.ylim([0.4, 0.55])
```

[14]:
```
(0.4, 0.55)
```



[15]:
```
print(rf_cv.best_params_)
```

```
{'max_features': 16, 'min_samples_leaf': 5, 'n_estimators': 40, 'random_state':
88}
```

2.2.3 Evaluate the model performance (trained on the entire training set)

[16]:
```
print('Cross-validated R2:', round(rf_cv.best_score_, 5))
print('OSR2:', round(OSR2(rf_cv, X_test, y_test, y_train), 5))
```

```
Cross-validated R2: 0.52633
OSR2: 0.55854
```

2.2.4 Feature Importance

[17]:
```
pd.DataFrame({'Feature' : X_train.columns,
              'Importance score': 100*rf_cv.best_estimator_.
 ↪feature_importances_}).round(1)
```

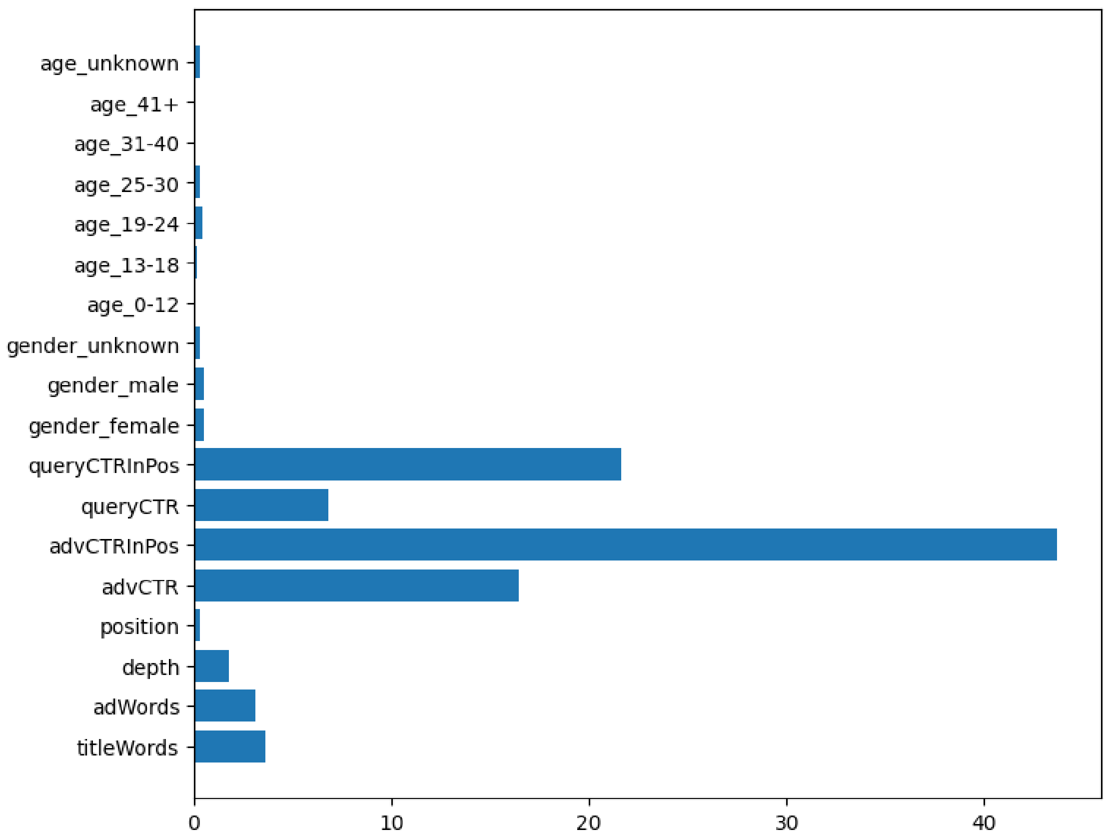
[17]:

	Feature	Importance score
0	titleWords	3.6
1	adWords	3.1
2	depth	1.8

```
3          position          0.3
4            advCTR         16.4
5       advCTRInPos         43.7
6          queryCTR          6.8
7     queryCTRInPos         21.6
8     gender_female          0.5
9       gender_male          0.5
10   gender_unknown          0.3
11          age_0-12         0.0
12         age_13-18         0.2
13         age_19-24         0.4
14         age_25-30         0.3
15         age_31-40         0.1
16           age_41+         0.0
17       age_unknown         0.3
```

[18]:

```
plt.figure(figsize=(8,7))
plt.barh(X_train.columns, 100*rf_cv.best_estimator_.feature_importances_)
plt.show()
```



1.3 3. GRADIENT BOOSTED TREES

1.3.1 3.1 Gradient Boosting Regressor

Controlling tree size using `max_leaf_nodes` vs. `max_depth`: https://scikit-learn.org/stable/modules/ensemble.html#controlling-the-tree-size

We choose use `max_leaf_nodes` as our primary parameter to control the size of the tree: we set `max_depth` to a large value so that it does not interfere with the construction of the trees.

[19]:
```
from sklearn.ensemble import GradientBoostingRegressor

gbr = GradientBoostingRegressor(n_estimators=2000, learning_rate= 0.001,␣
 ↪max_leaf_nodes=3,
                                max_depth=10, min_samples_leaf=10,␣
 ↪random_state=88, verbose=2)
gbr.fit(X_train, y_train)
```

```
      Iter       Train Loss   Remaining Time
         1           0.0060            2.21m
         2           0.0060            1.21m
         3           0.0060           51.93s
         4           0.0060           41.42s
         5           0.0060           35.11s
         6           0.0060           31.00s
         7           0.0060           27.97s
         8           0.0060           25.71s
         9           0.0060           23.96s
        10           0.0060           22.57s
        11           0.0060           21.45s
        12           0.0060           20.47s
        13           0.0060           19.65s
        14           0.0060           19.02s
        15           0.0060           18.42s
        16           0.0059           17.87s
        17           0.0059           17.41s
        18           0.0059           16.98s
        19           0.0059           16.64s
        20           0.0059           16.34s
        21           0.0059           16.04s
        22           0.0059           15.78s
        23           0.0059           15.56s
        24           0.0059           15.34s
        25           0.0059           15.15s
        26           0.0059           14.96s
        27           0.0059           14.78s
        28           0.0059           14.61s
        29           0.0059           14.46s
        30           0.0059           14.31s
```

31	0.0059	14.17s
32	0.0059	14.03s
33	0.0059	13.91s
34	0.0059	13.79s
35	0.0059	13.67s
36	0.0059	13.55s
37	0.0059	13.45s
38	0.0059	13.34s
39	0.0059	13.25s
40	0.0059	13.16s
41	0.0059	13.08s
42	0.0059	13.04s
43	0.0059	12.97s
44	0.0058	12.90s
45	0.0058	12.92s
46	0.0058	13.12s
47	0.0058	13.06s
48	0.0058	12.98s
49	0.0058	12.90s
50	0.0058	12.83s
51	0.0058	12.77s
52	0.0058	12.72s
53	0.0058	12.66s
54	0.0058	12.61s
55	0.0058	12.55s
56	0.0058	12.50s
57	0.0058	12.44s
58	0.0058	12.40s
59	0.0058	12.35s
60	0.0058	12.30s
61	0.0058	12.26s
62	0.0058	12.21s
63	0.0058	12.17s
64	0.0058	12.13s
65	0.0058	12.09s
66	0.0058	12.06s
67	0.0058	12.02s
68	0.0058	11.98s
69	0.0058	11.95s
70	0.0058	11.91s
71	0.0058	11.87s
72	0.0058	11.83s
73	0.0057	11.80s
74	0.0057	11.77s
75	0.0057	11.74s
76	0.0057	11.70s
77	0.0057	11.67s
78	0.0057	11.64s

79	0.0057	11.61s
80	0.0057	11.58s
81	0.0057	11.56s
82	0.0057	11.53s
83	0.0057	11.51s
84	0.0057	11.48s
85	0.0057	11.45s
86	0.0057	11.42s
87	0.0057	11.40s
88	0.0057	11.38s
89	0.0057	11.35s
90	0.0057	11.34s
91	0.0057	11.32s
92	0.0057	11.29s
93	0.0057	11.27s
94	0.0057	11.26s
95	0.0057	11.24s
96	0.0057	11.22s
97	0.0057	11.20s
98	0.0057	11.19s
99	0.0057	11.20s
100	0.0057	11.18s
101	0.0057	11.16s
102	0.0057	11.15s
103	0.0057	11.13s
104	0.0056	11.11s
105	0.0056	11.09s
106	0.0056	11.07s
107	0.0056	11.05s
108	0.0056	11.03s
109	0.0056	11.02s
110	0.0056	11.00s
111	0.0056	10.98s
112	0.0056	10.97s
113	0.0056	10.95s
114	0.0056	10.94s
115	0.0056	10.92s
116	0.0056	10.90s
117	0.0056	10.89s
118	0.0056	10.87s
119	0.0056	10.86s
120	0.0056	10.84s
121	0.0056	10.83s
122	0.0056	10.82s
123	0.0056	10.80s
124	0.0056	10.78s
125	0.0056	10.77s
126	0.0056	10.75s

127	0.0056	10.73s
128	0.0056	10.72s
129	0.0056	10.70s
130	0.0056	10.69s
131	0.0056	10.67s
132	0.0056	10.65s
133	0.0056	10.64s
134	0.0056	10.63s
135	0.0056	10.61s
136	0.0056	10.60s
137	0.0056	10.58s
138	0.0055	10.57s
139	0.0055	10.56s
140	0.0055	10.54s
141	0.0055	10.54s
142	0.0055	10.53s
143	0.0055	10.51s
144	0.0055	10.50s
145	0.0055	10.51s
146	0.0055	10.50s
147	0.0055	10.48s
148	0.0055	10.46s
149	0.0055	10.45s
150	0.0055	10.42s
151	0.0055	10.41s
152	0.0055	10.39s
153	0.0055	10.39s
154	0.0055	10.38s
155	0.0055	10.36s
156	0.0055	10.35s
157	0.0055	10.34s
158	0.0055	10.31s
159	0.0055	10.30s
160	0.0055	10.28s
161	0.0055	10.27s
162	0.0055	10.25s
163	0.0055	10.23s
164	0.0055	10.22s
165	0.0055	10.20s
166	0.0055	10.17s
167	0.0055	10.16s
168	0.0055	10.15s
169	0.0055	10.14s
170	0.0055	10.12s
171	0.0055	10.11s
172	0.0055	10.09s
173	0.0054	10.08s
174	0.0054	10.07s

175	0.0054	10.05s
176	0.0054	10.04s
177	0.0054	10.02s
178	0.0054	10.01s
179	0.0054	10.01s
180	0.0054	10.01s
181	0.0054	9.99s
182	0.0054	9.97s
183	0.0054	9.97s
184	0.0054	9.96s
185	0.0054	9.95s
186	0.0054	9.94s
187	0.0054	9.93s
188	0.0054	9.93s
189	0.0054	9.91s
190	0.0054	9.90s
191	0.0054	9.89s
192	0.0054	9.88s
193	0.0054	9.87s
194	0.0054	9.86s
195	0.0054	9.85s
196	0.0054	9.83s
197	0.0054	9.82s
198	0.0054	9.81s
199	0.0054	9.81s
200	0.0054	9.79s
201	0.0054	9.77s
202	0.0054	9.77s
203	0.0054	9.76s
204	0.0054	9.76s
205	0.0054	9.75s
206	0.0054	9.74s
207	0.0054	9.73s
208	0.0054	9.73s
209	0.0053	9.72s
210	0.0053	9.70s
211	0.0053	9.69s
212	0.0053	9.67s
213	0.0053	9.66s
214	0.0053	9.65s
215	0.0053	9.64s
216	0.0053	9.63s
217	0.0053	9.61s
218	0.0053	9.60s
219	0.0053	9.60s
220	0.0053	9.59s
221	0.0053	9.57s
222	0.0053	9.56s

223	0.0053	9.55s
224	0.0053	9.55s
225	0.0053	9.53s
226	0.0053	9.52s
227	0.0053	9.52s
228	0.0053	9.51s
229	0.0053	9.51s
230	0.0053	9.49s
231	0.0053	9.48s
232	0.0053	9.48s
233	0.0053	9.46s
234	0.0053	9.45s
235	0.0053	9.45s
236	0.0053	9.44s
237	0.0053	9.42s
238	0.0053	9.41s
239	0.0053	9.41s
240	0.0053	9.40s
241	0.0053	9.39s
242	0.0053	9.38s
243	0.0053	9.37s
244	0.0053	9.37s
245	0.0053	9.35s
246	0.0053	9.34s
247	0.0053	9.36s
248	0.0052	9.37s
249	0.0052	9.38s
250	0.0052	9.37s
251	0.0052	9.36s
252	0.0052	9.36s
253	0.0052	9.34s
254	0.0052	9.34s
255	0.0052	9.33s
256	0.0052	9.32s
257	0.0052	9.31s
258	0.0052	9.30s
259	0.0052	9.29s
260	0.0052	9.29s
261	0.0052	9.27s
262	0.0052	9.26s
263	0.0052	9.26s
264	0.0052	9.25s
265	0.0052	9.24s
266	0.0052	9.23s
267	0.0052	9.22s
268	0.0052	9.21s
269	0.0052	9.21s
270	0.0052	9.19s

271	0.0052	9.18s
272	0.0052	9.18s
273	0.0052	9.17s
274	0.0052	9.17s
275	0.0052	9.15s
276	0.0052	9.15s
277	0.0052	9.14s
278	0.0052	9.13s
279	0.0052	9.12s
280	0.0052	9.11s
281	0.0052	9.10s
282	0.0052	9.09s
283	0.0052	9.09s
284	0.0052	9.08s
285	0.0052	9.08s
286	0.0052	9.07s
287	0.0052	9.06s
288	0.0052	9.05s
289	0.0051	9.04s
290	0.0051	9.03s
291	0.0051	9.03s
292	0.0051	9.02s
293	0.0051	9.01s
294	0.0051	9.00s
295	0.0051	8.99s
296	0.0051	8.99s
297	0.0051	8.98s
298	0.0051	8.97s
299	0.0051	8.96s
300	0.0051	8.96s
301	0.0051	8.95s
302	0.0051	8.94s
303	0.0051	8.93s
304	0.0051	8.92s
305	0.0051	8.91s
306	0.0051	8.91s
307	0.0051	8.90s
308	0.0051	8.89s
309	0.0051	8.88s
310	0.0051	8.87s
311	0.0051	8.87s
312	0.0051	8.86s
313	0.0051	8.86s
314	0.0051	8.84s
315	0.0051	8.84s
316	0.0051	8.83s
317	0.0051	8.82s
318	0.0051	8.81s

319	0.0051	8.80s
320	0.0051	8.80s
321	0.0051	8.79s
322	0.0051	8.78s
323	0.0051	8.78s
324	0.0051	8.76s
325	0.0051	8.76s
326	0.0051	8.76s
327	0.0051	8.76s
328	0.0051	8.76s
329	0.0051	8.76s
330	0.0051	8.75s
331	0.0051	8.75s
332	0.0051	8.74s
333	0.0050	8.73s
334	0.0050	8.73s
335	0.0050	8.72s
336	0.0050	8.71s
337	0.0050	8.71s
338	0.0050	8.71s
339	0.0050	8.71s
340	0.0050	8.71s
341	0.0050	8.70s
342	0.0050	8.71s
343	0.0050	8.72s
344	0.0050	8.72s
345	0.0050	8.72s
346	0.0050	8.72s
347	0.0050	8.72s
348	0.0050	8.71s
349	0.0050	8.71s
350	0.0050	8.71s
351	0.0050	8.71s
352	0.0050	8.70s
353	0.0050	8.69s
354	0.0050	8.68s
355	0.0050	8.68s
356	0.0050	8.67s
357	0.0050	8.66s
358	0.0050	8.66s
359	0.0050	8.65s
360	0.0050	8.64s
361	0.0050	8.64s
362	0.0050	8.63s
363	0.0050	8.62s
364	0.0050	8.61s
365	0.0050	8.61s
366	0.0050	8.60s

367	0.0050	8.60s
368	0.0050	8.59s
369	0.0050	8.58s
370	0.0050	8.58s
371	0.0050	8.57s
372	0.0050	8.56s
373	0.0050	8.56s
374	0.0050	8.55s
375	0.0050	8.55s
376	0.0050	8.54s
377	0.0050	8.53s
378	0.0050	8.53s
379	0.0050	8.52s
380	0.0049	8.53s
381	0.0049	8.53s
382	0.0049	8.52s
383	0.0049	8.51s
384	0.0049	8.50s
385	0.0049	8.49s
386	0.0049	8.49s
387	0.0049	8.48s
388	0.0049	8.48s
389	0.0049	8.47s
390	0.0049	8.46s
391	0.0049	8.45s
392	0.0049	8.45s
393	0.0049	8.44s
394	0.0049	8.45s
395	0.0049	8.45s
396	0.0049	8.44s
397	0.0049	8.43s
398	0.0049	8.44s
399	0.0049	8.43s
400	0.0049	8.43s
401	0.0049	8.42s
402	0.0049	8.41s
403	0.0049	8.41s
404	0.0049	8.40s
405	0.0049	8.39s
406	0.0049	8.39s
407	0.0049	8.38s
408	0.0049	8.37s
409	0.0049	8.36s
410	0.0049	8.35s
411	0.0049	8.35s
412	0.0049	8.34s
413	0.0049	8.33s
414	0.0049	8.33s

415	0.0049	8.32s
416	0.0049	8.31s
417	0.0049	8.31s
418	0.0049	8.30s
419	0.0049	8.29s
420	0.0049	8.29s
421	0.0049	8.28s
422	0.0049	8.27s
423	0.0049	8.27s
424	0.0049	8.26s
425	0.0049	8.25s
426	0.0049	8.24s
427	0.0049	8.24s
428	0.0049	8.23s
429	0.0049	8.23s
430	0.0049	8.22s
431	0.0048	8.22s
432	0.0048	8.21s
433	0.0048	8.20s
434	0.0048	8.20s
435	0.0048	8.19s
436	0.0048	8.18s
437	0.0048	8.18s
438	0.0048	8.17s
439	0.0048	8.16s
440	0.0048	8.16s
441	0.0048	8.15s
442	0.0048	8.14s
443	0.0048	8.14s
444	0.0048	8.13s
445	0.0048	8.12s
446	0.0048	8.12s
447	0.0048	8.11s
448	0.0048	8.10s
449	0.0048	8.10s
450	0.0048	8.09s
451	0.0048	8.08s
452	0.0048	8.08s
453	0.0048	8.07s
454	0.0048	8.06s
455	0.0048	8.05s
456	0.0048	8.05s
457	0.0048	8.04s
458	0.0048	8.03s
459	0.0048	8.03s
460	0.0048	8.02s
461	0.0048	8.01s
462	0.0048	8.00s

463	0.0048	8.00s
464	0.0048	7.99s
465	0.0048	7.99s
466	0.0048	7.98s
467	0.0048	7.97s
468	0.0048	7.96s
469	0.0048	7.96s
470	0.0048	7.95s
471	0.0048	7.95s
472	0.0048	7.94s
473	0.0048	7.93s
474	0.0048	7.92s
475	0.0048	7.92s
476	0.0048	7.91s
477	0.0048	7.90s
478	0.0048	7.90s
479	0.0048	7.89s
480	0.0048	7.88s
481	0.0048	7.88s
482	0.0048	7.87s
483	0.0048	7.87s
484	0.0048	7.86s
485	0.0048	7.85s
486	0.0047	7.85s
487	0.0047	7.84s
488	0.0047	7.83s
489	0.0047	7.83s
490	0.0047	7.82s
491	0.0047	7.81s
492	0.0047	7.81s
493	0.0047	7.80s
494	0.0047	7.79s
495	0.0047	7.78s
496	0.0047	7.78s
497	0.0047	7.77s
498	0.0047	7.77s
499	0.0047	7.76s
500	0.0047	7.76s
501	0.0047	7.75s
502	0.0047	7.74s
503	0.0047	7.74s
504	0.0047	7.73s
505	0.0047	7.73s
506	0.0047	7.72s
507	0.0047	7.71s
508	0.0047	7.71s
509	0.0047	7.70s
510	0.0047	7.69s

511	0.0047	7.69s
512	0.0047	7.68s
513	0.0047	7.67s
514	0.0047	7.67s
515	0.0047	7.67s
516	0.0047	7.66s
517	0.0047	7.65s
518	0.0047	7.64s
519	0.0047	7.64s
520	0.0047	7.63s
521	0.0047	7.62s
522	0.0047	7.61s
523	0.0047	7.61s
524	0.0047	7.60s
525	0.0047	7.60s
526	0.0047	7.60s
527	0.0047	7.59s
528	0.0047	7.58s
529	0.0047	7.58s
530	0.0047	7.57s
531	0.0047	7.56s
532	0.0047	7.55s
533	0.0047	7.55s
534	0.0047	7.54s
535	0.0047	7.53s
536	0.0047	7.53s
537	0.0047	7.52s
538	0.0047	7.52s
539	0.0047	7.51s
540	0.0047	7.51s
541	0.0047	7.50s
542	0.0047	7.49s
543	0.0047	7.48s
544	0.0047	7.48s
545	0.0047	7.47s
546	0.0046	7.47s
547	0.0046	7.46s
548	0.0046	7.46s
549	0.0046	7.45s
550	0.0046	7.44s
551	0.0046	7.44s
552	0.0046	7.43s
553	0.0046	7.42s
554	0.0046	7.41s
555	0.0046	7.41s
556	0.0046	7.40s
557	0.0046	7.40s
558	0.0046	7.39s

559	0.0046	7.38s
560	0.0046	7.38s
561	0.0046	7.37s
562	0.0046	7.37s
563	0.0046	7.36s
564	0.0046	7.35s
565	0.0046	7.34s
566	0.0046	7.34s
567	0.0046	7.33s
568	0.0046	7.32s
569	0.0046	7.32s
570	0.0046	7.31s
571	0.0046	7.30s
572	0.0046	7.30s
573	0.0046	7.29s
574	0.0046	7.28s
575	0.0046	7.28s
576	0.0046	7.27s
577	0.0046	7.26s
578	0.0046	7.26s
579	0.0046	7.25s
580	0.0046	7.24s
581	0.0046	7.24s
582	0.0046	7.23s
583	0.0046	7.22s
584	0.0046	7.22s
585	0.0046	7.21s
586	0.0046	7.20s
587	0.0046	7.20s
588	0.0046	7.19s
589	0.0046	7.18s
590	0.0046	7.17s
591	0.0046	7.17s
592	0.0046	7.16s
593	0.0046	7.15s
594	0.0046	7.15s
595	0.0046	7.14s
596	0.0046	7.13s
597	0.0046	7.13s
598	0.0046	7.12s
599	0.0046	7.11s
600	0.0046	7.11s
601	0.0046	7.10s
602	0.0046	7.10s
603	0.0046	7.09s
604	0.0046	7.08s
605	0.0046	7.08s
606	0.0046	7.08s

607	0.0046	7.07s
608	0.0046	7.06s
609	0.0046	7.06s
610	0.0046	7.05s
611	0.0045	7.04s
612	0.0045	7.04s
613	0.0045	7.03s
614	0.0045	7.02s
615	0.0045	7.02s
616	0.0045	7.01s
617	0.0045	7.00s
618	0.0045	7.00s
619	0.0045	6.99s
620	0.0045	6.98s
621	0.0045	6.98s
622	0.0045	6.97s
623	0.0045	6.96s
624	0.0045	6.96s
625	0.0045	6.95s
626	0.0045	6.94s
627	0.0045	6.94s
628	0.0045	6.93s
629	0.0045	6.92s
630	0.0045	6.92s
631	0.0045	6.91s
632	0.0045	6.91s
633	0.0045	6.90s
634	0.0045	6.89s
635	0.0045	6.89s
636	0.0045	6.88s
637	0.0045	6.87s
638	0.0045	6.87s
639	0.0045	6.86s
640	0.0045	6.86s
641	0.0045	6.85s
642	0.0045	6.84s
643	0.0045	6.84s
644	0.0045	6.83s
645	0.0045	6.82s
646	0.0045	6.82s
647	0.0045	6.81s
648	0.0045	6.81s
649	0.0045	6.80s
650	0.0045	6.79s
651	0.0045	6.78s
652	0.0045	6.78s
653	0.0045	6.77s
654	0.0045	6.76s

655	0.0045	6.76s
656	0.0045	6.75s
657	0.0045	6.75s
658	0.0045	6.74s
659	0.0045	6.73s
660	0.0045	6.73s
661	0.0045	6.72s
662	0.0045	6.72s
663	0.0045	6.71s
664	0.0045	6.70s
665	0.0045	6.70s
666	0.0045	6.69s
667	0.0045	6.69s
668	0.0045	6.68s
669	0.0045	6.67s
670	0.0045	6.67s
671	0.0045	6.66s
672	0.0045	6.65s
673	0.0045	6.65s
674	0.0045	6.64s
675	0.0045	6.64s
676	0.0045	6.63s
677	0.0045	6.62s
678	0.0045	6.62s
679	0.0045	6.61s
680	0.0045	6.60s
681	0.0045	6.60s
682	0.0045	6.59s
683	0.0044	6.59s
684	0.0044	6.58s
685	0.0044	6.58s
686	0.0044	6.57s
687	0.0044	6.56s
688	0.0044	6.56s
689	0.0044	6.55s
690	0.0044	6.55s
691	0.0044	6.54s
692	0.0044	6.53s
693	0.0044	6.53s
694	0.0044	6.52s
695	0.0044	6.52s
696	0.0044	6.51s
697	0.0044	6.50s
698	0.0044	6.50s
699	0.0044	6.49s
700	0.0044	6.49s
701	0.0044	6.48s
702	0.0044	6.47s

703	0.0044	6.47s
704	0.0044	6.46s
705	0.0044	6.46s
706	0.0044	6.45s
707	0.0044	6.45s
708	0.0044	6.44s
709	0.0044	6.44s
710	0.0044	6.43s
711	0.0044	6.42s
712	0.0044	6.42s
713	0.0044	6.41s
714	0.0044	6.40s
715	0.0044	6.40s
716	0.0044	6.39s
717	0.0044	6.39s
718	0.0044	6.38s
719	0.0044	6.38s
720	0.0044	6.37s
721	0.0044	6.37s
722	0.0044	6.36s
723	0.0044	6.36s
724	0.0044	6.35s
725	0.0044	6.35s
726	0.0044	6.34s
727	0.0044	6.33s
728	0.0044	6.33s
729	0.0044	6.32s
730	0.0044	6.32s
731	0.0044	6.31s
732	0.0044	6.31s
733	0.0044	6.30s
734	0.0044	6.30s
735	0.0044	6.29s
736	0.0044	6.28s
737	0.0044	6.28s
738	0.0044	6.27s
739	0.0044	6.27s
740	0.0044	6.26s
741	0.0044	6.26s
742	0.0044	6.25s
743	0.0044	6.25s
744	0.0044	6.24s
745	0.0044	6.24s
746	0.0044	6.23s
747	0.0044	6.22s
748	0.0044	6.22s
749	0.0044	6.21s
750	0.0044	6.21s

751	0.0044	6.20s
752	0.0044	6.20s
753	0.0044	6.19s
754	0.0044	6.19s
755	0.0044	6.18s
756	0.0044	6.18s
757	0.0044	6.17s
758	0.0044	6.17s
759	0.0044	6.16s
760	0.0044	6.15s
761	0.0044	6.15s
762	0.0044	6.14s
763	0.0044	6.14s
764	0.0043	6.13s
765	0.0043	6.13s
766	0.0043	6.12s
767	0.0043	6.12s
768	0.0043	6.11s
769	0.0043	6.11s
770	0.0043	6.10s
771	0.0043	6.10s
772	0.0043	6.09s
773	0.0043	6.08s
774	0.0043	6.08s
775	0.0043	6.07s
776	0.0043	6.07s
777	0.0043	6.06s
778	0.0043	6.06s
779	0.0043	6.05s
780	0.0043	6.04s
781	0.0043	6.04s
782	0.0043	6.03s
783	0.0043	6.03s
784	0.0043	6.02s
785	0.0043	6.02s
786	0.0043	6.01s
787	0.0043	6.01s
788	0.0043	6.00s
789	0.0043	6.00s
790	0.0043	5.99s
791	0.0043	5.98s
792	0.0043	5.98s
793	0.0043	5.97s
794	0.0043	5.97s
795	0.0043	5.96s
796	0.0043	5.96s
797	0.0043	5.95s
798	0.0043	5.94s

799	0.0043	5.94s
800	0.0043	5.93s
801	0.0043	5.93s
802	0.0043	5.92s
803	0.0043	5.92s
804	0.0043	5.91s
805	0.0043	5.91s
806	0.0043	5.90s
807	0.0043	5.89s
808	0.0043	5.89s
809	0.0043	5.88s
810	0.0043	5.88s
811	0.0043	5.87s
812	0.0043	5.87s
813	0.0043	5.86s
814	0.0043	5.86s
815	0.0043	5.85s
816	0.0043	5.84s
817	0.0043	5.84s
818	0.0043	5.83s
819	0.0043	5.83s
820	0.0043	5.82s
821	0.0043	5.82s
822	0.0043	5.81s
823	0.0043	5.81s
824	0.0043	5.80s
825	0.0043	5.80s
826	0.0043	5.79s
827	0.0043	5.78s
828	0.0043	5.78s
829	0.0043	5.77s
830	0.0043	5.77s
831	0.0043	5.76s
832	0.0043	5.76s
833	0.0043	5.75s
834	0.0043	5.75s
835	0.0043	5.74s
836	0.0043	5.74s
837	0.0043	5.73s
838	0.0043	5.73s
839	0.0043	5.72s
840	0.0043	5.71s
841	0.0043	5.71s
842	0.0043	5.70s
843	0.0043	5.70s
844	0.0043	5.69s
845	0.0043	5.69s
846	0.0043	5.68s

847	0.0043	5.68s
848	0.0043	5.67s
849	0.0043	5.66s
850	0.0043	5.66s
851	0.0043	5.65s
852	0.0043	5.65s
853	0.0042	5.64s
854	0.0042	5.64s
855	0.0042	5.63s
856	0.0042	5.63s
857	0.0042	5.62s
858	0.0042	5.62s
859	0.0042	5.61s
860	0.0042	5.61s
861	0.0042	5.60s
862	0.0042	5.60s
863	0.0042	5.59s
864	0.0042	5.58s
865	0.0042	5.58s
866	0.0042	5.57s
867	0.0042	5.57s
868	0.0042	5.56s
869	0.0042	5.56s
870	0.0042	5.55s
871	0.0042	5.55s
872	0.0042	5.54s
873	0.0042	5.54s
874	0.0042	5.53s
875	0.0042	5.53s
876	0.0042	5.52s
877	0.0042	5.51s
878	0.0042	5.51s
879	0.0042	5.50s
880	0.0042	5.50s
881	0.0042	5.49s
882	0.0042	5.49s
883	0.0042	5.48s
884	0.0042	5.48s
885	0.0042	5.47s
886	0.0042	5.47s
887	0.0042	5.46s
888	0.0042	5.46s
889	0.0042	5.45s
890	0.0042	5.45s
891	0.0042	5.44s
892	0.0042	5.44s
893	0.0042	5.43s
894	0.0042	5.43s

895	0.0042	5.42s
896	0.0042	5.42s
897	0.0042	5.41s
898	0.0042	5.41s
899	0.0042	5.40s
900	0.0042	5.40s
901	0.0042	5.39s
902	0.0042	5.38s
903	0.0042	5.38s
904	0.0042	5.37s
905	0.0042	5.37s
906	0.0042	5.36s
907	0.0042	5.36s
908	0.0042	5.35s
909	0.0042	5.35s
910	0.0042	5.34s
911	0.0042	5.34s
912	0.0042	5.33s
913	0.0042	5.33s
914	0.0042	5.32s
915	0.0042	5.32s
916	0.0042	5.31s
917	0.0042	5.31s
918	0.0042	5.30s
919	0.0042	5.30s
920	0.0042	5.29s
921	0.0042	5.29s
922	0.0042	5.28s
923	0.0042	5.28s
924	0.0042	5.27s
925	0.0042	5.27s
926	0.0042	5.26s
927	0.0042	5.26s
928	0.0042	5.25s
929	0.0042	5.25s
930	0.0042	5.24s
931	0.0042	5.24s
932	0.0042	5.23s
933	0.0042	5.23s
934	0.0042	5.22s
935	0.0042	5.22s
936	0.0042	5.21s
937	0.0042	5.21s
938	0.0042	5.20s
939	0.0042	5.20s
940	0.0042	5.19s
941	0.0042	5.19s
942	0.0042	5.18s

943	0.0042	5.17s
944	0.0042	5.17s
945	0.0042	5.16s
946	0.0042	5.16s
947	0.0042	5.15s
948	0.0042	5.15s
949	0.0042	5.14s
950	0.0042	5.14s
951	0.0042	5.13s
952	0.0042	5.13s
953	0.0042	5.12s
954	0.0041	5.12s
955	0.0041	5.11s
956	0.0041	5.11s
957	0.0041	5.10s
958	0.0041	5.10s
959	0.0041	5.09s
960	0.0041	5.09s
961	0.0041	5.08s
962	0.0041	5.08s
963	0.0041	5.08s
964	0.0041	5.07s
965	0.0041	5.06s
966	0.0041	5.06s
967	0.0041	5.05s
968	0.0041	5.05s
969	0.0041	5.04s
970	0.0041	5.04s
971	0.0041	5.04s
972	0.0041	5.03s
973	0.0041	5.02s
974	0.0041	5.02s
975	0.0041	5.01s
976	0.0041	5.01s
977	0.0041	5.00s
978	0.0041	5.00s
979	0.0041	4.99s
980	0.0041	4.99s
981	0.0041	4.98s
982	0.0041	4.98s
983	0.0041	4.97s
984	0.0041	4.97s
985	0.0041	4.96s
986	0.0041	4.96s
987	0.0041	4.95s
988	0.0041	4.95s
989	0.0041	4.94s
990	0.0041	4.94s

991	0.0041	4.93s
992	0.0041	4.93s
993	0.0041	4.92s
994	0.0041	4.92s
995	0.0041	4.92s
996	0.0041	4.91s
997	0.0041	4.90s
998	0.0041	4.90s
999	0.0041	4.89s
1000	0.0041	4.89s
1001	0.0041	4.89s
1002	0.0041	4.88s
1003	0.0041	4.88s
1004	0.0041	4.87s
1005	0.0041	4.87s
1006	0.0041	4.86s
1007	0.0041	4.86s
1008	0.0041	4.85s
1009	0.0041	4.85s
1010	0.0041	4.84s
1011	0.0041	4.83s
1012	0.0041	4.83s
1013	0.0041	4.82s
1014	0.0041	4.82s
1015	0.0041	4.81s
1016	0.0041	4.81s
1017	0.0041	4.80s
1018	0.0041	4.80s
1019	0.0041	4.79s
1020	0.0041	4.79s
1021	0.0041	4.78s
1022	0.0041	4.78s
1023	0.0041	4.77s
1024	0.0041	4.77s
1025	0.0041	4.76s
1026	0.0041	4.76s
1027	0.0041	4.75s
1028	0.0041	4.75s
1029	0.0041	4.74s
1030	0.0041	4.74s
1031	0.0041	4.73s
1032	0.0041	4.73s
1033	0.0041	4.72s
1034	0.0041	4.72s
1035	0.0041	4.71s
1036	0.0041	4.71s
1037	0.0041	4.70s
1038	0.0041	4.70s

1039	0.0041	4.69s
1040	0.0041	4.69s
1041	0.0041	4.68s
1042	0.0041	4.68s
1043	0.0041	4.67s
1044	0.0041	4.67s
1045	0.0041	4.66s
1046	0.0041	4.66s
1047	0.0041	4.65s
1048	0.0041	4.65s
1049	0.0041	4.64s
1050	0.0041	4.64s
1051	0.0041	4.63s
1052	0.0041	4.63s
1053	0.0041	4.62s
1054	0.0041	4.62s
1055	0.0041	4.61s
1056	0.0041	4.61s
1057	0.0041	4.60s
1058	0.0041	4.60s
1059	0.0041	4.59s
1060	0.0041	4.59s
1061	0.0041	4.58s
1062	0.0041	4.57s
1063	0.0041	4.57s
1064	0.0041	4.57s
1065	0.0040	4.56s
1066	0.0040	4.56s
1067	0.0040	4.55s
1068	0.0040	4.55s
1069	0.0040	4.54s
1070	0.0040	4.54s
1071	0.0040	4.53s
1072	0.0040	4.53s
1073	0.0040	4.52s
1074	0.0040	4.52s
1075	0.0040	4.51s
1076	0.0040	4.51s
1077	0.0040	4.50s
1078	0.0040	4.50s
1079	0.0040	4.49s
1080	0.0040	4.49s
1081	0.0040	4.48s
1082	0.0040	4.48s
1083	0.0040	4.47s
1084	0.0040	4.47s
1085	0.0040	4.46s
1086	0.0040	4.46s

1087	0.0040	4.45s
1088	0.0040	4.45s
1089	0.0040	4.44s
1090	0.0040	4.44s
1091	0.0040	4.43s
1092	0.0040	4.43s
1093	0.0040	4.42s
1094	0.0040	4.42s
1095	0.0040	4.41s
1096	0.0040	4.41s
1097	0.0040	4.40s
1098	0.0040	4.40s
1099	0.0040	4.39s
1100	0.0040	4.39s
1101	0.0040	4.38s
1102	0.0040	4.38s
1103	0.0040	4.37s
1104	0.0040	4.37s
1105	0.0040	4.36s
1106	0.0040	4.36s
1107	0.0040	4.35s
1108	0.0040	4.35s
1109	0.0040	4.34s
1110	0.0040	4.34s
1111	0.0040	4.33s
1112	0.0040	4.33s
1113	0.0040	4.32s
1114	0.0040	4.32s
1115	0.0040	4.31s
1116	0.0040	4.31s
1117	0.0040	4.30s
1118	0.0040	4.30s
1119	0.0040	4.29s
1120	0.0040	4.29s
1121	0.0040	4.28s
1122	0.0040	4.28s
1123	0.0040	4.27s
1124	0.0040	4.27s
1125	0.0040	4.26s
1126	0.0040	4.26s
1127	0.0040	4.25s
1128	0.0040	4.25s
1129	0.0040	4.24s
1130	0.0040	4.24s
1131	0.0040	4.23s
1132	0.0040	4.23s
1133	0.0040	4.22s
1134	0.0040	4.22s

1135	0.0040	4.21s
1136	0.0040	4.21s
1137	0.0040	4.20s
1138	0.0040	4.20s
1139	0.0040	4.19s
1140	0.0040	4.19s
1141	0.0040	4.18s
1142	0.0040	4.18s
1143	0.0040	4.17s
1144	0.0040	4.17s
1145	0.0040	4.16s
1146	0.0040	4.16s
1147	0.0040	4.16s
1148	0.0040	4.15s
1149	0.0040	4.15s
1150	0.0040	4.14s
1151	0.0040	4.13s
1152	0.0040	4.13s
1153	0.0040	4.13s
1154	0.0040	4.12s
1155	0.0040	4.11s
1156	0.0040	4.11s
1157	0.0040	4.11s
1158	0.0040	4.10s
1159	0.0040	4.10s
1160	0.0040	4.09s
1161	0.0040	4.09s
1162	0.0040	4.08s
1163	0.0040	4.08s
1164	0.0040	4.07s
1165	0.0040	4.07s
1166	0.0040	4.06s
1167	0.0040	4.06s
1168	0.0040	4.05s
1169	0.0040	4.05s
1170	0.0040	4.04s
1171	0.0040	4.04s
1172	0.0040	4.03s
1173	0.0040	4.03s
1174	0.0040	4.02s
1175	0.0040	4.02s
1176	0.0040	4.01s
1177	0.0040	4.01s
1178	0.0040	4.00s
1179	0.0040	4.00s
1180	0.0040	3.99s
1181	0.0040	3.99s
1182	0.0040	3.98s

1183	0.0040	3.98s
1184	0.0040	3.97s
1185	0.0040	3.97s
1186	0.0040	3.96s
1187	0.0040	3.96s
1188	0.0040	3.95s
1189	0.0040	3.95s
1190	0.0039	3.94s
1191	0.0039	3.94s
1192	0.0039	3.93s
1193	0.0039	3.93s
1194	0.0039	3.92s
1195	0.0039	3.92s
1196	0.0039	3.91s
1197	0.0039	3.91s
1198	0.0039	3.91s
1199	0.0039	3.90s
1200	0.0039	3.90s
1201	0.0039	3.89s
1202	0.0039	3.89s
1203	0.0039	3.88s
1204	0.0039	3.88s
1205	0.0039	3.87s
1206	0.0039	3.87s
1207	0.0039	3.86s
1208	0.0039	3.86s
1209	0.0039	3.85s
1210	0.0039	3.85s
1211	0.0039	3.85s
1212	0.0039	3.84s
1213	0.0039	3.84s
1214	0.0039	3.83s
1215	0.0039	3.83s
1216	0.0039	3.82s
1217	0.0039	3.82s
1218	0.0039	3.81s
1219	0.0039	3.81s
1220	0.0039	3.80s
1221	0.0039	3.80s
1222	0.0039	3.79s
1223	0.0039	3.79s
1224	0.0039	3.78s
1225	0.0039	3.78s
1226	0.0039	3.77s
1227	0.0039	3.77s
1228	0.0039	3.76s
1229	0.0039	3.76s
1230	0.0039	3.75s

1231	0.0039	3.75s
1232	0.0039	3.74s
1233	0.0039	3.74s
1234	0.0039	3.73s
1235	0.0039	3.73s
1236	0.0039	3.72s
1237	0.0039	3.72s
1238	0.0039	3.71s
1239	0.0039	3.71s
1240	0.0039	3.70s
1241	0.0039	3.70s
1242	0.0039	3.69s
1243	0.0039	3.69s
1244	0.0039	3.68s
1245	0.0039	3.68s
1246	0.0039	3.67s
1247	0.0039	3.67s
1248	0.0039	3.66s
1249	0.0039	3.66s
1250	0.0039	3.65s
1251	0.0039	3.65s
1252	0.0039	3.64s
1253	0.0039	3.64s
1254	0.0039	3.64s
1255	0.0039	3.63s
1256	0.0039	3.63s
1257	0.0039	3.62s
1258	0.0039	3.62s
1259	0.0039	3.61s
1260	0.0039	3.61s
1261	0.0039	3.60s
1262	0.0039	3.60s
1263	0.0039	3.59s
1264	0.0039	3.59s
1265	0.0039	3.58s
1266	0.0039	3.58s
1267	0.0039	3.57s
1268	0.0039	3.57s
1269	0.0039	3.56s
1270	0.0039	3.56s
1271	0.0039	3.55s
1272	0.0039	3.55s
1273	0.0039	3.54s
1274	0.0039	3.54s
1275	0.0039	3.53s
1276	0.0039	3.53s
1277	0.0039	3.52s
1278	0.0039	3.52s

1279	0.0039	3.51s
1280	0.0039	3.51s
1281	0.0039	3.50s
1282	0.0039	3.50s
1283	0.0039	3.49s
1284	0.0039	3.49s
1285	0.0039	3.48s
1286	0.0039	3.48s
1287	0.0039	3.47s
1288	0.0039	3.47s
1289	0.0039	3.46s
1290	0.0039	3.46s
1291	0.0039	3.45s
1292	0.0039	3.45s
1293	0.0039	3.44s
1294	0.0039	3.44s
1295	0.0039	3.43s
1296	0.0039	3.43s
1297	0.0039	3.42s
1298	0.0039	3.42s
1299	0.0039	3.42s
1300	0.0039	3.41s
1301	0.0039	3.41s
1302	0.0039	3.40s
1303	0.0039	3.40s
1304	0.0039	3.39s
1305	0.0039	3.39s
1306	0.0039	3.38s
1307	0.0039	3.38s
1308	0.0039	3.37s
1309	0.0039	3.37s
1310	0.0039	3.36s
1311	0.0039	3.36s
1312	0.0039	3.35s
1313	0.0039	3.35s
1314	0.0039	3.34s
1315	0.0039	3.34s
1316	0.0039	3.33s
1317	0.0039	3.33s
1318	0.0039	3.32s
1319	0.0039	3.32s
1320	0.0039	3.31s
1321	0.0039	3.31s
1322	0.0039	3.30s
1323	0.0039	3.30s
1324	0.0039	3.29s
1325	0.0039	3.29s
1326	0.0039	3.28s

1327	0.0039	3.28s
1328	0.0039	3.27s
1329	0.0039	3.27s
1330	0.0039	3.26s
1331	0.0039	3.26s
1332	0.0039	3.26s
1333	0.0039	3.25s
1334	0.0039	3.25s
1335	0.0039	3.24s
1336	0.0038	3.24s
1337	0.0038	3.23s
1338	0.0038	3.23s
1339	0.0038	3.22s
1340	0.0038	3.22s
1341	0.0038	3.21s
1342	0.0038	3.21s
1343	0.0038	3.20s
1344	0.0038	3.20s
1345	0.0038	3.19s
1346	0.0038	3.19s
1347	0.0038	3.18s
1348	0.0038	3.18s
1349	0.0038	3.17s
1350	0.0038	3.17s
1351	0.0038	3.16s
1352	0.0038	3.16s
1353	0.0038	3.15s
1354	0.0038	3.15s
1355	0.0038	3.14s
1356	0.0038	3.14s
1357	0.0038	3.14s
1358	0.0038	3.13s
1359	0.0038	3.13s
1360	0.0038	3.12s
1361	0.0038	3.12s
1362	0.0038	3.11s
1363	0.0038	3.11s
1364	0.0038	3.10s
1365	0.0038	3.10s
1366	0.0038	3.09s
1367	0.0038	3.09s
1368	0.0038	3.08s
1369	0.0038	3.08s
1370	0.0038	3.07s
1371	0.0038	3.07s
1372	0.0038	3.06s
1373	0.0038	3.06s
1374	0.0038	3.05s

1375	0.0038	3.05s
1376	0.0038	3.04s
1377	0.0038	3.04s
1378	0.0038	3.03s
1379	0.0038	3.03s
1380	0.0038	3.02s
1381	0.0038	3.02s
1382	0.0038	3.01s
1383	0.0038	3.01s
1384	0.0038	3.00s
1385	0.0038	3.00s
1386	0.0038	2.99s
1387	0.0038	2.99s
1388	0.0038	2.98s
1389	0.0038	2.98s
1390	0.0038	2.97s
1391	0.0038	2.97s
1392	0.0038	2.97s
1393	0.0038	2.96s
1394	0.0038	2.96s
1395	0.0038	2.95s
1396	0.0038	2.95s
1397	0.0038	2.94s
1398	0.0038	2.94s
1399	0.0038	2.93s
1400	0.0038	2.93s
1401	0.0038	2.92s
1402	0.0038	2.92s
1403	0.0038	2.91s
1404	0.0038	2.91s
1405	0.0038	2.90s
1406	0.0038	2.90s
1407	0.0038	2.90s
1408	0.0038	2.89s
1409	0.0038	2.89s
1410	0.0038	2.88s
1411	0.0038	2.88s
1412	0.0038	2.87s
1413	0.0038	2.87s
1414	0.0038	2.86s
1415	0.0038	2.86s
1416	0.0038	2.85s
1417	0.0038	2.85s
1418	0.0038	2.84s
1419	0.0038	2.84s
1420	0.0038	2.83s
1421	0.0038	2.83s
1422	0.0038	2.82s

1423	0.0038	2.82s
1424	0.0038	2.81s
1425	0.0038	2.81s
1426	0.0038	2.81s
1427	0.0038	2.80s
1428	0.0038	2.80s
1429	0.0038	2.79s
1430	0.0038	2.79s
1431	0.0038	2.78s
1432	0.0038	2.78s
1433	0.0038	2.77s
1434	0.0038	2.77s
1435	0.0038	2.76s
1436	0.0038	2.76s
1437	0.0038	2.75s
1438	0.0038	2.75s
1439	0.0038	2.74s
1440	0.0038	2.74s
1441	0.0038	2.73s
1442	0.0038	2.73s
1443	0.0038	2.72s
1444	0.0038	2.72s
1445	0.0038	2.71s
1446	0.0038	2.71s
1447	0.0038	2.70s
1448	0.0038	2.70s
1449	0.0038	2.69s
1450	0.0038	2.69s
1451	0.0038	2.68s
1452	0.0038	2.68s
1453	0.0038	2.67s
1454	0.0038	2.67s
1455	0.0038	2.66s
1456	0.0038	2.66s
1457	0.0038	2.65s
1458	0.0038	2.65s
1459	0.0038	2.64s
1460	0.0038	2.64s
1461	0.0038	2.63s
1462	0.0038	2.63s
1463	0.0038	2.62s
1464	0.0038	2.62s
1465	0.0038	2.62s
1466	0.0038	2.61s
1467	0.0038	2.61s
1468	0.0038	2.60s
1469	0.0038	2.60s
1470	0.0038	2.59s

1471	0.0038	2.59s
1472	0.0038	2.58s
1473	0.0038	2.58s
1474	0.0038	2.57s
1475	0.0038	2.57s
1476	0.0038	2.56s
1477	0.0038	2.56s
1478	0.0038	2.55s
1479	0.0038	2.55s
1480	0.0038	2.54s
1481	0.0038	2.54s
1482	0.0038	2.53s
1483	0.0038	2.53s
1484	0.0038	2.52s
1485	0.0038	2.52s
1486	0.0038	2.51s
1487	0.0038	2.51s
1488	0.0038	2.50s
1489	0.0038	2.50s
1490	0.0038	2.49s
1491	0.0038	2.49s
1492	0.0038	2.48s
1493	0.0038	2.48s
1494	0.0038	2.47s
1495	0.0038	2.47s
1496	0.0038	2.46s
1497	0.0038	2.46s
1498	0.0038	2.45s
1499	0.0038	2.45s
1500	0.0038	2.44s
1501	0.0038	2.44s
1502	0.0038	2.43s
1503	0.0038	2.43s
1504	0.0037	2.42s
1505	0.0037	2.42s
1506	0.0037	2.41s
1507	0.0037	2.41s
1508	0.0037	2.41s
1509	0.0037	2.40s
1510	0.0037	2.40s
1511	0.0037	2.39s
1512	0.0037	2.39s
1513	0.0037	2.38s
1514	0.0037	2.38s
1515	0.0037	2.37s
1516	0.0037	2.37s
1517	0.0037	2.36s
1518	0.0037	2.36s

1519	0.0037	2.35s
1520	0.0037	2.35s
1521	0.0037	2.34s
1522	0.0037	2.34s
1523	0.0037	2.33s
1524	0.0037	2.33s
1525	0.0037	2.32s
1526	0.0037	2.32s
1527	0.0037	2.31s
1528	0.0037	2.31s
1529	0.0037	2.30s
1530	0.0037	2.30s
1531	0.0037	2.29s
1532	0.0037	2.29s
1533	0.0037	2.28s
1534	0.0037	2.28s
1535	0.0037	2.27s
1536	0.0037	2.27s
1537	0.0037	2.26s
1538	0.0037	2.26s
1539	0.0037	2.25s
1540	0.0037	2.25s
1541	0.0037	2.24s
1542	0.0037	2.24s
1543	0.0037	2.23s
1544	0.0037	2.23s
1545	0.0037	2.22s
1546	0.0037	2.22s
1547	0.0037	2.21s
1548	0.0037	2.21s
1549	0.0037	2.20s
1550	0.0037	2.20s
1551	0.0037	2.19s
1552	0.0037	2.19s
1553	0.0037	2.18s
1554	0.0037	2.18s
1555	0.0037	2.18s
1556	0.0037	2.17s
1557	0.0037	2.17s
1558	0.0037	2.16s
1559	0.0037	2.16s
1560	0.0037	2.15s
1561	0.0037	2.15s
1562	0.0037	2.14s
1563	0.0037	2.14s
1564	0.0037	2.13s
1565	0.0037	2.13s
1566	0.0037	2.12s

1567	0.0037	2.12s
1568	0.0037	2.11s
1569	0.0037	2.11s
1570	0.0037	2.10s
1571	0.0037	2.10s
1572	0.0037	2.09s
1573	0.0037	2.09s
1574	0.0037	2.08s
1575	0.0037	2.08s
1576	0.0037	2.07s
1577	0.0037	2.07s
1578	0.0037	2.06s
1579	0.0037	2.06s
1580	0.0037	2.05s
1581	0.0037	2.05s
1582	0.0037	2.04s
1583	0.0037	2.04s
1584	0.0037	2.03s
1585	0.0037	2.03s
1586	0.0037	2.02s
1587	0.0037	2.02s
1588	0.0037	2.01s
1589	0.0037	2.01s
1590	0.0037	2.00s
1591	0.0037	2.00s
1592	0.0037	1.99s
1593	0.0037	1.99s
1594	0.0037	1.98s
1595	0.0037	1.98s
1596	0.0037	1.97s
1597	0.0037	1.97s
1598	0.0037	1.97s
1599	0.0037	1.96s
1600	0.0037	1.96s
1601	0.0037	1.95s
1602	0.0037	1.95s
1603	0.0037	1.94s
1604	0.0037	1.94s
1605	0.0037	1.93s
1606	0.0037	1.93s
1607	0.0037	1.92s
1608	0.0037	1.92s
1609	0.0037	1.91s
1610	0.0037	1.91s
1611	0.0037	1.90s
1612	0.0037	1.90s
1613	0.0037	1.89s
1614	0.0037	1.89s

1615	0.0037	1.88s
1616	0.0037	1.88s
1617	0.0037	1.87s
1618	0.0037	1.87s
1619	0.0037	1.86s
1620	0.0037	1.86s
1621	0.0037	1.85s
1622	0.0037	1.85s
1623	0.0037	1.84s
1624	0.0037	1.84s
1625	0.0037	1.83s
1626	0.0037	1.83s
1627	0.0037	1.82s
1628	0.0037	1.82s
1629	0.0037	1.81s
1630	0.0037	1.81s
1631	0.0037	1.80s
1632	0.0037	1.80s
1633	0.0037	1.79s
1634	0.0037	1.79s
1635	0.0037	1.78s
1636	0.0037	1.78s
1637	0.0037	1.77s
1638	0.0037	1.77s
1639	0.0037	1.76s
1640	0.0037	1.76s
1641	0.0037	1.75s
1642	0.0037	1.75s
1643	0.0037	1.74s
1644	0.0037	1.74s
1645	0.0037	1.73s
1646	0.0037	1.73s
1647	0.0037	1.72s
1648	0.0037	1.72s
1649	0.0037	1.71s
1650	0.0037	1.71s
1651	0.0037	1.70s
1652	0.0037	1.70s
1653	0.0037	1.69s
1654	0.0037	1.69s
1655	0.0037	1.68s
1656	0.0037	1.68s
1657	0.0037	1.67s
1658	0.0037	1.67s
1659	0.0037	1.66s
1660	0.0037	1.66s
1661	0.0037	1.65s
1662	0.0037	1.65s

1663	0.0037	1.64s
1664	0.0037	1.64s
1665	0.0037	1.63s
1666	0.0037	1.63s
1667	0.0037	1.62s
1668	0.0037	1.62s
1669	0.0037	1.61s
1670	0.0037	1.61s
1671	0.0037	1.60s
1672	0.0037	1.60s
1673	0.0037	1.59s
1674	0.0037	1.59s
1675	0.0037	1.58s
1676	0.0037	1.58s
1677	0.0037	1.57s
1678	0.0037	1.57s
1679	0.0037	1.57s
1680	0.0037	1.57s
1681	0.0037	1.56s
1682	0.0037	1.56s
1683	0.0037	1.55s
1684	0.0037	1.55s
1685	0.0037	1.54s
1686	0.0037	1.54s
1687	0.0037	1.53s
1688	0.0037	1.53s
1689	0.0037	1.52s
1690	0.0037	1.52s
1691	0.0037	1.51s
1692	0.0037	1.51s
1693	0.0037	1.50s
1694	0.0037	1.50s
1695	0.0037	1.49s
1696	0.0037	1.49s
1697	0.0037	1.48s
1698	0.0037	1.48s
1699	0.0037	1.47s
1700	0.0037	1.47s
1701	0.0037	1.46s
1702	0.0037	1.46s
1703	0.0037	1.45s
1704	0.0037	1.45s
1705	0.0037	1.44s
1706	0.0037	1.44s
1707	0.0037	1.43s
1708	0.0037	1.43s
1709	0.0036	1.42s
1710	0.0036	1.42s

1711	0.0036	1.41s
1712	0.0036	1.41s
1713	0.0036	1.40s
1714	0.0036	1.40s
1715	0.0036	1.39s
1716	0.0036	1.39s
1717	0.0036	1.38s
1718	0.0036	1.38s
1719	0.0036	1.37s
1720	0.0036	1.37s
1721	0.0036	1.36s
1722	0.0036	1.36s
1723	0.0036	1.35s
1724	0.0036	1.35s
1725	0.0036	1.34s
1726	0.0036	1.34s
1727	0.0036	1.33s
1728	0.0036	1.33s
1729	0.0036	1.32s
1730	0.0036	1.32s
1731	0.0036	1.31s
1732	0.0036	1.31s
1733	0.0036	1.30s
1734	0.0036	1.30s
1735	0.0036	1.29s
1736	0.0036	1.29s
1737	0.0036	1.28s
1738	0.0036	1.28s
1739	0.0036	1.27s
1740	0.0036	1.27s
1741	0.0036	1.26s
1742	0.0036	1.26s
1743	0.0036	1.25s
1744	0.0036	1.25s
1745	0.0036	1.24s
1746	0.0036	1.24s
1747	0.0036	1.23s
1748	0.0036	1.23s
1749	0.0036	1.22s
1750	0.0036	1.22s
1751	0.0036	1.21s
1752	0.0036	1.21s
1753	0.0036	1.20s
1754	0.0036	1.20s
1755	0.0036	1.19s
1756	0.0036	1.19s
1757	0.0036	1.18s
1758	0.0036	1.18s

1759	0.0036	1.17s
1760	0.0036	1.17s
1761	0.0036	1.16s
1762	0.0036	1.16s
1763	0.0036	1.15s
1764	0.0036	1.15s
1765	0.0036	1.14s
1766	0.0036	1.14s
1767	0.0036	1.13s
1768	0.0036	1.13s
1769	0.0036	1.12s
1770	0.0036	1.12s
1771	0.0036	1.11s
1772	0.0036	1.11s
1773	0.0036	1.10s
1774	0.0036	1.10s
1775	0.0036	1.10s
1776	0.0036	1.09s
1777	0.0036	1.09s
1778	0.0036	1.08s
1779	0.0036	1.08s
1780	0.0036	1.07s
1781	0.0036	1.07s
1782	0.0036	1.06s
1783	0.0036	1.06s
1784	0.0036	1.05s
1785	0.0036	1.05s
1786	0.0036	1.04s
1787	0.0036	1.04s
1788	0.0036	1.03s
1789	0.0036	1.03s
1790	0.0036	1.02s
1791	0.0036	1.02s
1792	0.0036	1.01s
1793	0.0036	1.01s
1794	0.0036	1.00s
1795	0.0036	1.00s
1796	0.0036	0.99s
1797	0.0036	0.99s
1798	0.0036	0.98s
1799	0.0036	0.98s
1800	0.0036	0.97s
1801	0.0036	0.97s
1802	0.0036	0.96s
1803	0.0036	0.96s
1804	0.0036	0.95s
1805	0.0036	0.95s
1806	0.0036	0.94s

1807	0.0036	0.94s
1808	0.0036	0.93s
1809	0.0036	0.93s
1810	0.0036	0.92s
1811	0.0036	0.92s
1812	0.0036	0.91s
1813	0.0036	0.91s
1814	0.0036	0.91s
1815	0.0036	0.90s
1816	0.0036	0.90s
1817	0.0036	0.89s
1818	0.0036	0.89s
1819	0.0036	0.88s
1820	0.0036	0.88s
1821	0.0036	0.87s
1822	0.0036	0.87s
1823	0.0036	0.86s
1824	0.0036	0.86s
1825	0.0036	0.85s
1826	0.0036	0.85s
1827	0.0036	0.84s
1828	0.0036	0.84s
1829	0.0036	0.83s
1830	0.0036	0.83s
1831	0.0036	0.82s
1832	0.0036	0.82s
1833	0.0036	0.81s
1834	0.0036	0.81s
1835	0.0036	0.80s
1836	0.0036	0.80s
1837	0.0036	0.80s
1838	0.0036	0.79s
1839	0.0036	0.79s
1840	0.0036	0.78s
1841	0.0036	0.78s
1842	0.0036	0.77s
1843	0.0036	0.77s
1844	0.0036	0.76s
1845	0.0036	0.76s
1846	0.0036	0.75s
1847	0.0036	0.75s
1848	0.0036	0.74s
1849	0.0036	0.74s
1850	0.0036	0.73s
1851	0.0036	0.73s
1852	0.0036	0.72s
1853	0.0036	0.72s
1854	0.0036	0.71s

1855	0.0036	0.71s
1856	0.0036	0.70s
1857	0.0036	0.70s
1858	0.0036	0.69s
1859	0.0036	0.69s
1860	0.0036	0.68s
1861	0.0036	0.68s
1862	0.0036	0.67s
1863	0.0036	0.67s
1864	0.0036	0.66s
1865	0.0036	0.66s
1866	0.0036	0.65s
1867	0.0036	0.65s
1868	0.0036	0.64s
1869	0.0036	0.64s
1870	0.0036	0.63s
1871	0.0036	0.63s
1872	0.0036	0.62s
1873	0.0036	0.62s
1874	0.0036	0.61s
1875	0.0036	0.61s
1876	0.0036	0.60s
1877	0.0036	0.60s
1878	0.0036	0.59s
1879	0.0036	0.59s
1880	0.0036	0.59s
1881	0.0036	0.58s
1882	0.0036	0.58s
1883	0.0036	0.57s
1884	0.0036	0.57s
1885	0.0036	0.56s
1886	0.0036	0.56s
1887	0.0036	0.55s
1888	0.0036	0.55s
1889	0.0036	0.54s
1890	0.0036	0.54s
1891	0.0036	0.53s
1892	0.0036	0.53s
1893	0.0036	0.52s
1894	0.0036	0.52s
1895	0.0036	0.51s
1896	0.0036	0.51s
1897	0.0036	0.50s
1898	0.0036	0.50s
1899	0.0036	0.49s
1900	0.0036	0.49s
1901	0.0036	0.48s
1902	0.0036	0.48s

1903	0.0036	0.47s
1904	0.0036	0.47s
1905	0.0036	0.46s
1906	0.0036	0.46s
1907	0.0036	0.45s
1908	0.0036	0.45s
1909	0.0036	0.44s
1910	0.0036	0.44s
1911	0.0036	0.43s
1912	0.0036	0.43s
1913	0.0036	0.42s
1914	0.0036	0.42s
1915	0.0036	0.41s
1916	0.0036	0.41s
1917	0.0036	0.40s
1918	0.0036	0.40s
1919	0.0036	0.39s
1920	0.0036	0.39s
1921	0.0036	0.38s
1922	0.0036	0.38s
1923	0.0036	0.37s
1924	0.0036	0.37s
1925	0.0036	0.37s
1926	0.0036	0.36s
1927	0.0036	0.36s
1928	0.0036	0.35s
1929	0.0036	0.35s
1930	0.0036	0.34s
1931	0.0036	0.34s
1932	0.0036	0.33s
1933	0.0036	0.33s
1934	0.0036	0.32s
1935	0.0036	0.32s
1936	0.0036	0.31s
1937	0.0036	0.31s
1938	0.0036	0.30s
1939	0.0036	0.30s
1940	0.0036	0.29s
1941	0.0036	0.29s
1942	0.0036	0.28s
1943	0.0036	0.28s
1944	0.0036	0.27s
1945	0.0036	0.27s
1946	0.0036	0.26s
1947	0.0036	0.26s
1948	0.0036	0.25s
1949	0.0036	0.25s
1950	0.0036	0.24s

1951	0.0036	0.24s
1952	0.0036	0.23s
1953	0.0035	0.23s
1954	0.0035	0.22s
1955	0.0035	0.22s
1956	0.0035	0.21s
1957	0.0035	0.21s
1958	0.0035	0.20s
1959	0.0035	0.20s
1960	0.0035	0.19s
1961	0.0035	0.19s
1962	0.0035	0.18s
1963	0.0035	0.18s
1964	0.0035	0.18s
1965	0.0035	0.17s
1966	0.0035	0.17s
1967	0.0035	0.16s
1968	0.0035	0.16s
1969	0.0035	0.15s
1970	0.0035	0.15s
1971	0.0035	0.14s
1972	0.0035	0.14s
1973	0.0035	0.13s
1974	0.0035	0.13s
1975	0.0035	0.12s
1976	0.0035	0.12s
1977	0.0035	0.11s
1978	0.0035	0.11s
1979	0.0035	0.10s
1980	0.0035	0.10s
1981	0.0035	0.09s
1982	0.0035	0.09s
1983	0.0035	0.08s
1984	0.0035	0.08s
1985	0.0035	0.07s
1986	0.0035	0.07s
1987	0.0035	0.06s
1988	0.0035	0.06s
1989	0.0035	0.05s
1990	0.0035	0.05s
1991	0.0035	0.04s
1992	0.0035	0.04s
1993	0.0035	0.03s
1994	0.0035	0.03s
1995	0.0035	0.02s
1996	0.0035	0.02s
1997	0.0035	0.01s
1998	0.0035	0.01s

```
1999            0.0035             0.00s
2000            0.0035             0.00s
```

[19]:
```
GradientBoostingRegressor(learning_rate=0.001, max_depth=10, max_leaf_nodes=3,
                          min_samples_leaf=10, n_estimators=2000,
                          random_state=88, verbose=2)
```

[20]:
```python
print('OSR2:', round(OSR2(gbr, X_test, y_test, y_train), 5))
```

```
OSR2: 0.4388
```

Note that the OSR2 of the above gbr model is not very good. It is because the model is quite sensitive to hypterparameters. We will learn how to find the best parameters using cross-validation and see the improvement one could get from choosing the best hyperparameters.

1.3.2 3.2 Gradient Boosting Regressor with CV

Note that if you use the `GridSearchCV` function directly with the gbr as the classifier, the run-time is super long (e.g., more than 30 hours). The reason is that the naive implement of `GridSearchCV` does not take into account the special additive nature of the Boosting models.

For example, if you need to decide what is the best number of trees to include in your model and you would like to try values from 1 to 10000. The naive implementation of `GridSearchCV` would train 10000 different gbr models. However, a much more efficient way is to train a model with 10000 trees, and only subset a subset of them in your model when needed. This trick allows us to reduce the training time of gbr_cv to less than 2 hours.

If you use the `caret` package in `R`, they have implemented this efficient algorithm, but there is not a good counterpart in `Python sklearn`.

However, we change these values to a small subset to reduce the run-time.

1.1 Split the training data into 5 folds for cross validation

[21]:
```python
grid_values = {'n_estimators': np.arange(3000, 7000, 20)}

gbr = GradientBoostingRegressor(min_samples_leaf=5, n_estimators=120,␣
 ↪min_samples_split=20, random_state=88)

cv = KFold(n_splits=5,random_state=1,shuffle=True)

gbr_cv = GridSearchCV(gbr, param_grid=grid_values, scoring='r2', cv=cv,␣
 ↪verbose=1)
gbr_cv.fit(X_train, y_train)
```

```
Fitting 5 folds for each of 200 candidates, totalling 1000 fits
```

```
/Users/ivynangalia/Library/Python/3.12/lib/python/site-
packages/numpy/ma/core.py:2820: RuntimeWarning: invalid value encountered in
cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

[21]:
```
GridSearchCV(cv=KFold(n_splits=5, random_state=1, shuffle=True),
             estimator=GradientBoostingRegressor(min_samples_leaf=5,
                                                 min_samples_split=20,
                                                 n_estimators=120,
                                                 random_state=88),
             param_grid={'n_estimators': array([3000, 3020, 3040, 3060, 3080,
3100, 3120, 3140, 3160, 3180, 3200,
       3220, 3240, 3260, 3280, 3300, 3320, 3340, 3360, 3380, 3400, 3420,
       3440, 3460, 3480, 3500, 3520, 3540, 3560…
       5640, 5660, 5680, 5700, 5720, 5740, 5760, 5780, 5800, 5820, 5840,
       5860, 5880, 5900, 5920, 5940, 5960, 5980, 6000, 6020, 6040, 6060,
       6080, 6100, 6120, 6140, 6160, 6180, 6200, 6220, 6240, 6260, 6280,
       6300, 6320, 6340, 6360, 6380, 6400, 6420, 6440, 6460, 6480, 6500,
       6520, 6540, 6560, 6580, 6600, 6620, 6640, 6660, 6680, 6700, 6720,
       6740, 6760, 6780, 6800, 6820, 6840, 6860, 6880, 6900, 6920, 6940,
       6960, 6980])},
             scoring='r2', verbose=1)
```

[ ]:
```

```

1.6 Train the model with the best hyperparameters on the full training data.

[58]:
```
gbr_cv = GradientBoostingRegressor(n_estimators = 5000, learning_rate = 0.005,␣
 ↪max_depth = 20,
                                   max_leaf_nodes=7,␣
 ↪min_samples_leaf=10,random_state=99,verbose=1)
gbr_cv.fit(X_train,y_train)
```

```
      Iter       Train Loss   Remaining Time
         1           0.0060            1.31m
         2           0.0060           54.35s
         3           0.0059           45.81s
         4           0.0059           41.34s
         5           0.0059           38.82s
         6           0.0059           36.53s
         7           0.0058           34.81s
         8           0.0058           33.55s
         9           0.0058           32.57s
        10           0.0058           31.54s
        20           0.0055           26.63s
        30           0.0053           24.98s
        40           0.0051           23.99s
        50           0.0050           24.07s
        60           0.0048           23.90s
        70           0.0047           23.70s
        80           0.0045           23.77s
        90           0.0044           23.73s
       100           0.0043           23.67s
```

```
     200           0.0035           24.50s
     300           0.0031           24.04s
     400           0.0029           23.41s
     500           0.0028           23.11s
     600           0.0026           22.95s
     700           0.0026           22.15s
     800           0.0025           21.45s
     900           0.0024           20.99s
    1000           0.0024           20.61s
    2000           0.0021           15.26s
    3000           0.0019           10.00s
    4000           0.0017            4.96s
    5000           0.0016            0.00s
```

[58]:
```
GradientBoostingRegressor(learning_rate=0.005, max_depth=20, max_leaf_nodes=7,
                          min_samples_leaf=10, n_estimators=5000,
                          random_state=99, verbose=1)
```

1.7 Evaluate the full model

[54]:
```python
print('OSR2:', round(OSR2(gbr_cv, X_test, y_test, y_train), 5))
```

```
OSR2: 0.53653
```

[55]:
```python
pd.DataFrame({'Feature' : X_train.columns,
              'Importance score': 100*gbr_cv.feature_importances_}).round(1)
```

[55]:

	Feature	Importance score
0	titleWords	1.7
1	adWords	2.0
2	depth	2.4
3	position	0.4
4	advCTR	13.2
5	advCTRInPos	52.6
6	queryCTR	4.8
7	queryCTRInPos	21.4
8	gender_female	0.5
9	gender_male	0.2
10	gender_unknown	0.2
11	age_0-12	0.0
12	age_13-18	0.0
13	age_19-24	0.1
14	age_25-30	0.2
15	age_31-40	0.0
16	age_41+	0.0
17	age_unknown	0.3

1.4 4. Final Comparison

1.4.1 4.1 Linear Regression Baseline

[56]:
```
from sklearn.linear_model import LinearRegression
import sklearn.metrics as metrics

lr = LinearRegression().fit(X_train, y_train)
```

1.4.2 4.2 Comparison Table

[62]:
```
comparison_data = {'Linear Regression': ['{:.3f}'.format(OSR2(lr, X_test,␣
 ↪y_test, y_train)),
                                         '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test, lr.predict(X_test))),
                                         '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test, lr.predict(X_test)))],
                   'Decision Tree Regressor': ['{:.3f}'.format(OSR2(dtr_cv,␣
 ↪X_test, y_test, y_train)),
                                               '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test, dtr_cv.predict(X_test))),
                                               '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test, dtr_cv.predict(X_test)))],
                   'Random Forest': ['{:.3f}'.format(OSR2(rf_cv, X_test,␣
 ↪y_test, y_train)),
                                     '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test, rf_cv.predict(X_test))),
                                     '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test, rf_cv.predict(X_test)))],
                   'Gradient Boosted Trees': ['{:.3f}'.format(OSR2(gbr_cv,␣
 ↪X_test, y_test, y_train)),
                                              '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test, gbr_cv.predict(X_test))),
                                              '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test, gbr_cv.predict(X_test)))]}

comparison_table = pd.DataFrame(data=comparison_data, index=['OSR2',␣
 ↪'Out-of-sample MSE', 'Out-of-sample MAE'])
comparison_table.style.set_properties(**{'font-size': '12pt',}).
 ↪set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}])
```

[62]:
```
<pandas.io.formats.style.Styler at 0x16ad2e9c0>
```

Let’s look at MAE restricted to CTR above and below 10%

1.5 In-class activity: Check if the prediction accuracy are different for low CTR and high CTR items. Use CTR <=0.1 or CTR > 0.1 as a threshold.

- Specifically, split the testing set to two subsets by CTR values. Evaluate the performance on each subset.

[59]:
```
X_test_low = X_test[y_test <= 0.1]
y_test_low = y_test[y_test <= 0.1]
```

[63]:
```
comparison_data = {'Linear Regression': ['{:.3f}'.format(OSR2(lr, X_test_low,␣
 ↪y_test_low, y_train)),
                                          '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test_low, lr.predict(X_test_low))),
                                          '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test_low, lr.predict(X_test_low)))],
                    'Decision Tree Regressor': ['{:.3f}'.format(OSR2(dtr_cv,␣
 ↪X_test_low, y_test_low, y_train)),
                                                '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test_low, dtr_cv.predict(X_test_low))),
                                                '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test_low, dtr_cv.predict(X_test_low)))],
                    'Random Forest': ['{:.3f}'.format(OSR2(rf_cv, X_test_low,␣
 ↪y_test_low, y_train)),
                                      '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test_low, rf_cv.predict(X_test_low))),
                                      '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test_low, rf_cv.predict(X_test_low)))],
                    'Gradient Boosted Trees': ['{:.3f}'.format(OSR2(gbr_cv,␣
 ↪X_test_low, y_test_low, y_train)),
                                               '{:.4f}'.format(metrics.
 ↪mean_squared_error(y_test_low, gbr_cv.predict(X_test_low))),
                                               '{:.3f}'.format(metrics.
 ↪mean_absolute_error(y_test_low, gbr_cv.predict(X_test_low)))]}

comparison_table = pd.DataFrame(data=comparison_data, index=['OSR2',␣
 ↪'Out-of-sample MSE', 'Out-of-sample MAE'])
comparison_table.style.set_properties(**{'font-size': '12pt',}).
 ↪set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}])
```

[63]:
```
<pandas.io.formats.style.Styler at 0x16ad2ede0>
```

[39]:
```
X_test_high = X_test[y_test >= 0.1]
y_test_high = y_test[y_test >= 0.1]
```

[64]:
```
comparison_data = {'Linear Regression': ['{:.3f}'.format(OSR2(lr, X_test_high,␣
 ↪y_test_high, y_train)),
```

```
                                         '{:.4f}'.format(metrics.
↪mean_squared_error(y_test_high, lr.predict(X_test_high))),
                                         '{:.3f}'.format(metrics.
↪mean_absolute_error(y_test_high, lr.predict(X_test_high)))],
                   'Decision Tree Regressor': ['{:.3f}'.format(OSR2(dtr_cv,␣
↪X_test_high, y_test_high, y_train)),
                                               '{:.4f}'.format(metrics.
↪mean_squared_error(y_test_high, dtr_cv.predict(X_test_high))),
                                               '{:.3f}'.format(metrics.
↪mean_absolute_error(y_test_high, dtr_cv.predict(X_test_high)))],
                   'Random Forest': ['{:.3f}'.format(OSR2(rf_cv, X_test_high,␣
↪y_test_high, y_train)),
                                     '{:.4f}'.format(metrics.
↪mean_squared_error(y_test_high, rf_cv.predict(X_test_high))),
                                     '{:.3f}'.format(metrics.
↪mean_absolute_error(y_test_high, rf_cv.predict(X_test_high)))],
                   'Gradient Boosted Trees': ['{:.3f}'.format(OSR2(gbr_cv,␣
↪X_test_high, y_test_high, y_train)),
                                              '{:.4f}'.format(metrics.
↪mean_squared_error(y_test_high, gbr_cv.predict(X_test_high))),
                                              '{:.3f}'.format(metrics.
↪mean_absolute_error(y_test_high, gbr_cv.predict(X_test_high)))]}

comparison_table = pd.DataFrame(data=comparison_data, index=['OSR2',␣
↪'Out-of-sample MSE', 'Out-of-sample MAE'])
comparison_table.style.set_properties(**{'font-size': '12pt',}).
↪set_table_styles([{'selector': 'th', 'props': [('font-size', '10pt')]}])
```

[64]: <pandas.io.formats.style.Styler at 0x16aef3320>