In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
df_data=pd.read_excel('data01.xlsx')
df_data.head(10)
```

Out[2]:

	Название площадки	Тип координат	Код объекта	Наименование	Срок выполнения работ	
0	Тверская область	line	A-TBP_M3_27	Установка и содержание направляющих ограждений...	03.10.2019 - 10.10.2022	N56.869465806346966,E3
1	Тверская область	line	A-TBP_M3_28	Установка и содержание направляющих ограждений...	03.10.2019 - 10.10.2022	N56.884591420817976,E3
2	Тверская область	line	A-TBP_M3_29	Установка и содержание направляющих ограждений	03.10.2019 - 08.11.2022	N56.87433246632805,E3
3	Тверская область	line	A-TBP_M3_30	Установка и содержание направляющих ограждений...	03.10.2019 - 10.10.2022	N56.847962835674466,E3
4	Тверская область	line	A-TBP_M3_36	Сахаровское ш. - ул. М. Василевского	15.05.2020 - 30.09.2020	N56.89850611693451,E36
5	Тверская область	line	A-TBP_M3_37	ул. Шишкова - ул. Старобежецкая	15.05.2020 - 30.09.2020	N56.881189988867625,E3
6	Тверская область	line	A-TBP_M3_38	проезд от ул.Дачная до дороги на подстанцию в ...	15.05.2020 - 30.09.2020	N56.89068340159709,E35
7	Тверская область	line	A-TBP_M3_39	ул. Маяковского	28.06.2020 - 30.09.2020	N56.87516176712185,E35
8	Тверская область	line	A-TBP_M3_40	Третьяковский пер.	03.08.2020 - 30.09.2020	N56.87514363232206,E35
9	Тверская область	line	A-TBP_M3_41	ул. Кольцевая	26.05.2020 - 30.09.2020	N56.8906859776843,E35.

In [3]:

```
df_data.shape
```

Out[3]:

```
(103, 6)
```

In [4]:

```
df_data.isnull().sum()
```

Out[4]:

```
Название площадки        0
Тип координат            0
Код объекта              0
Наименование             0
Срок выполнения работ    0
Координаты               0
dtype: int64
```

In [ ]:

```

```

In [5]:

```
import re
TOKEN_RE = re.compile(r'[\w\d]+')  #regular expression to start with
#tokenizer=WordPunctTokenizer()  #regular expr is better
```

In [6]:

```
def tokenize_text_simple_regex(txt, min_token_size=4):
    """ This func tokenize text with TOKEN_RE applied ealier """
    txt = txt.lower()
    all_tokens = TOKEN_RE.findall(txt)
    #all_tokens=tokenizer.tokenize(txt)
    all_tokens = [token for token in all_tokens if len(token) >= min_token_size]
    s=' '.join(all_tokens)
    return s
```

In [7]:

```
def concat_features(a,b,c,d):
  """
  Let s concat feature in order to make tfidf later
  """
  buff=(str(a)+' '+str(b)+' '+str(c)+' '+str(d))
  return buff
```

In [8]:

```
X_text=[]  #list for concat results
#loop on dataframe
for t in df_data.itertuples():
  buff=concat_features(t[3],t[4],t[5],t[6])
  buff=tokenize_text_simple_regex(buff, min_token_size=4)
  X_text.append(buff)
```

In [9]:

```
X_text[0]
```

Out[9]:

```
'твр_мз_27 установка содержание направляющих ограждений территории города
твери 2019 2022 869465806346966 87956666946412 86756001389617 892484188079
84 86754242151344 892602205276496 86566585316709 90823411941529 8654605977
9145 90999364852906 86464542960777 91716051101685'
```

In [11]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.metrics import f1_score
RS=42
```

In [12]:

```
X=TfidfVectorizer(
    tokenizer=tokenize_text_simple_regex,max_df=0.95,min_df=3
).fit_transform(X_text)
```

In [13]:

```
print(X)
```

```
  (0, 6)        0.05669735232523338
  (0, 15)       0.10135424906880018
  (0, 14)       0.16354007936895146
  (0, 0)        0.4290433565681024
  (0, 11)       0.250035847346694
  (0, 12)       0.3401841139514003
  (0, 9)        0.07424295500420514
  (0, 3)        0.25082039395901895
  (0, 4)        0.309483845228284
  (0, 5)        0.21695892829203584
  (0, 7)        0.49858772546498126
  (0, 13)       0.08851408097596429
  (0, 10)       0.09511109076076804
  (0, 27)       0.0845508495264477
  (0, 26)       0.17410012399296512
  (0, 21)       0.19457825559628336
  (0, 17)       0.09751655903832712
  (0, 2)        0.1599954850210816
  (0, 8)        0.10135424906880018
  (1, 6)        0.05669735232523338
  (1, 15)       0.10135424906880018
  (1, 14)       0.16354007936895146
  (1, 0)        0.4290433565681024
  (1, 11)       0.250035847346694
  (1, 12)       0.3401841139514003
  :     :
  (101, 3)      0.10530679545273994
  (101, 4)      0.23388589997698941
  (101, 5)      0.27327023679375995
  (101, 7)      0.2691408399728091
  (101, 10)     0.2395940143926491
  (101, 2)      0.1007610159716308
  (101, 8)      0.1276605662699591
  (101, 20)     0.15689318478814465
  (101, 1)      0.13663511839687997
  (101, 24)     0.1426981654318689
  (102, 14)     0.11686571263717144
  (102, 0)      0.17519674796382365
  (102, 11)     0.17867557358271666
  (102, 12)     0.8103183647331834
  (102, 9)      0.21221601159386833
  (102, 3)      0.11949080697145144
  (102, 4)      0.08846284106466462
  (102, 5)      0.15503881290557497
  (102, 7)      0.2035946874340639
  (102, 10)     0.1359327382540834
  (102, 2)      0.11433274612479158
  (102, 8)      0.14485545796401855
  (102, 20)     0.17802548428197068
  (102, 1)      0.15503881290557497
  (102, 24)     0.16191850551992126
```

In [14]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [15]:

```
sklearn_pipeline = Pipeline((('vect', TfidfVectorizer(tokenizer=tokenize_text_simple_re
gex,
                                                      max_df=0.95,
                                                      min_df=1)),
                             ('cls',
                             RandomForestClassifier(n_estimators=300,max_depth=25,n_job
s=-1, \
                                                    class_weight= 'balanced',verbose=True
,random_state=42))))
```

In [ ]:

```
sklearn_pipeline.fit(train_df['Vopros'], train_df['Otvet'])
sklearn_pipeline.score(train_df['Vopros'], train_df['Otvet'])
sklearn_pipeline.predict(train_df['Vopros'][0:10])
```

In [ ]:

```

```