In [1]:

```python
#爬虫部分
import time
import csv
import requests


def getArticleId(id_str):
    """
    :param id_str: 需要解密的id字符串
    :return:
    """
    url_id = "https://weibo.com/ajax/statuses/show?id={}".format(id_str)
    resp_id = requests.get(url_id, headers=headers)
    article_id = resp_id.json()["id"]
    return article_id


def get_one_page(params):
    """
    :param params: get请求需要的参数，数据类型为字典
    :return: max_id：请求所需的另一个参数
    """
    url = "https://weibo.com/ajax/statuses/buildComments"
    resp = requests.get(url, headers=headers, params=params)
    data_list = resp.json()["data"]
    for data in data_list:
        data_dict = {
            "screen_name": data["user"]["screen_name"],
            "location": data["user"]["location"],
            "created_time": data["created_at"].replace("+0800", ""),
            "text": data["text_raw"],
        }
        print(
            f'昵称：{data_dict["screen_name"]}\n地址：{data_dict["location"]}\n发布时间：{data_dict["created_time"]}\n评论内容：{data_dict[
        print("=" * 90)
        saveData(data_dict)
    max_id = resp.json()["max_id"]
    if max_id:
        return max_id
    else:
        return


def get_all_data(params):
    """
    :param params: get请求需要的参数，数据类型为字典
    :return:
    """
    max_id = get_one_page(params)
    params["max_id"] = max_id
    params["count"] = 20
    while max_id:
        params["max_id"] = max_id
        time.sleep(.5)
        max_id = get_one_page(params)


def saveData(data_dict):
    """
    :param data_dict: 要保存的数据，形式为dict类型
    :return:
    """
    writer.writerow(data_dict)


if __name__ == '__main__':
    uid = input("请输入作者id：")
    id_str = input("请输入您要爬取的微博话题的英文id：")
    fileName = input("请输入要保存的文件名：")
    headers = {
        "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) "
                      "Chrome/108.0.0.0 Safari/537.36 Edg/108.0.1462.46 ",
        "x-requested-with": "XMLHttpRequest",
        "referer": "https://weibo.com/1779719003/{}".format(id_str),
        "cookie": "SCF=AvbssvkbiAp5tHOZjddQU50gxRlXyswMPcCg-GgnR_PPY4e3dmLa_yzuu_xVl30OBY4SpE_XJRl1dPmba6gCOVk.; SUB=_2A25Ijh-NDeRhGeBN4lsF
        "x-xsrf-token": "-YYOKoKzkyMDGhDmhVSCLqpD"
    }

    id = getArticleId(id_str)  # 获取参数需要的真正id

    # 向csv文件写入表头
    header = ["screen_name","location", "created_time", "text"]
    f = open(f"{fileName}.csv", "w", encoding="utf-8", newline="")
    writer = csv.DictWriter(f, header)
    writer.writeheader()

    # get请求的参数
    params = {
        "is_reload": 1,
        "id": id,
        "is_show_bulletin": 2,
        "is_mix": 0,
        "count": 10,
        "uid": int(uid)
    }

    get_all_data(params)
    f.close()
    print("数据爬取完毕。")
```

```
请输入作者id：1642512402
请输入您要爬取的微博话题的英文id：4982906413909129
请输入要保存的文件名：final
昵称：改个啥名哈
地址：香港 其他
发布时间：Mon Dec 25 19:39:59  2023
评论内容：到底是有什么业务不办不行非得那天办
==========================================================================================
昵称：大鱼乐土
地址：河南 濮阳
发布时间：Mon Dec 25 18:54:25  2023
评论内容：除夕是非常重要的传统节日
==========================================================================================
昵称：定有趣
地址：浙江 杭州
发布时间：Mon Dec 25 19:47:36  2023
评论内容：除夕真的很重要，中国人除夕比初二，初三还要重要。
==========================================================================================
昵称：整条街最香的葱
```

1、导入数据以及相对应的库

In [21]:
```
!pip install snownlp -i  https://pypi.douban.com/simple
```

```
During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\utils\logging.py", line 177, in emit
    self.console.print(renderable, overflow="ignore", crop=False, style=style)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\console.py", line 1720, in print
    self._buffer.extend(new_segments)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\console.py", line 864, in __exit__
    self._exit_buffer()
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\console.py", line 822, in _exit_buffer
    self._check_buffer()
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\console.py", line 2027, in _check_buffer
    legacy_windows_render(buffer, LegacyWindowsTerm(self.file))
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\_windows_renderer.py", line 19, in legacy_windows_render
    term.write_text(text)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\rich\_win32_console.py", line 403, in write_text
    self.write(text)
UnicodeEncodeError: 'gbk' codec can't encode character '\xa0' in position 20: illegal multibyte sequence
```

In [5]:
```
import pandas as pd
import numpy as np
import re
import jieba
import jieba.posseg as pseg
from snownlp import SnowNLP
```

In [6]:
```
df = pd.read_csv('final.csv')
df.head()
```

Out[6]:

	screen_name	location	created_time	text
0	锦鲤六宝	其他	Wed Oct 25 10:27:46 2023	四大发明：调休、公摊、996、灵活就业 [doge]
1	我才不想跟你说话	重庆	Wed Oct 25 10:18:58 2023	一天到晚消费消费消费，就盯着老百姓口袋里的一点生活费[赞][赞][赞]
2	小武嘞	其他	Wed Oct 25 10:29:11 2023	除夕回不了家＝欢乐祥和？？？你怕是不知道有多少人是背井离乡在外上班
3	coconut_65	北京 海淀区	Wed Oct 25 10:29:15 2023	怎么不建议大家辞职呢
4	让羊再肥一会儿	四川 成都	Wed Oct 25 10:20:57 2023	消费个卵

2、做数据预处理，加入停用词，去掉无效词和标点符号以及分词处理

In [7]:
```
# 导入停用词列表
stop_words = []
with open("stopwords_cn.txt", 'r', encoding='utf-8') as f:
    lines = f.readlines()
    for line in lines:
        stop_words.append(line.strip())
stop_words
```

```
 ':',
 '://',
 '::',
 ';',
 '<',
 '=',
 '>',
 '>>',
 '?',
 '@',
 'Lex',
 '[',
 '\\',
 ']',
 '^',
 '_',
 '`',
 'exp',
 'sub',
 'sup'
```

In [8]:
```
#去掉标点符号，以及机械压缩
def preprocess_word(word):
    word1 = str(word)
    # word1 = re.sub(r'转发微博', '', word1)
    word1 = re.sub(r'#\w+#', '', word1)
    word1 = re.sub(r'【.*?】', '', word1)
    word1 = re.sub(r'@[\w]+', '', word1)
    word1 = re.sub(r'[a-zA-Z]', '', word1)
    word1 = re.sub(r'\.\d+', '', word1)
    return word1


def emjio_tihuan(x):
    x1 = str(x)
    x2 = re.sub('(\[.*?\])', "", x1)
    x3 = re.sub(r'@[\w\u2E80-\u9FFF]+:?|\[\w+\]', '', x2)
    x4 = re.sub(r'\n', '', x3)
    return x4

# 判断是否为中文
def is_all_chinese(strs):
    for _char in strs:
        if not '\u4e00' <= _char <= '\u9fa5':
            return False
    return True


# 定义机械压缩函数
def yasuo(st):
    for i in range(1, int(len(st) / 2) + 1):
        for j in range(len(st)):
            if st[j:j + i] == st[j + i:j + 2 * i]:
                k = j + i
                while st[k:k + i] == st[k + i:k + 2 * i] and k < len(st):
                    k = k + i
                st = st[:j] + st[k:]
    return st
```

In [9]:
```
def get_cut_words(content_series):
    try:
        # 对文本进行分词和词性标注
        words = pseg.cut(content_series)
        # 保存名词和形容词的列表
        nouns_and_adjs = []
        # 逐一检查每个词语的词性，并将名词和形容词保存到列表中
        for word, flag in words:
            if word not in stop_words and len(word) >= 2 and is_all_chinese(word) == True:
                # 如果是名词或形容词，就将其保存到列表中
                nouns_and_adjs.append(word)
        if len(nouns_and_adjs) != 0:
            return ' '.join(nouns_and_adjs)
        else:
            return np.NAN
    except:
        return np.NAN
```

```
In  [10]: df['text'] = df['text'].apply(preprocess_word)
          df['text']
```

```
Out[10]: 0                               四大发明：调休、公摊、996、灵活就业 []
         1                     一天到晚消费消费消费，就盯着老百姓口袋里的一点生活费[赞][赞][赞]
         2                      除夕回不了家＝欢乐祥和？？？你怕是不知道有多少人是背井离乡在外上班
         3                                         怎么不建议大家辞职呢
         4                                            消费个卵
                                        ...
         545    年轻人都去大城市，割断了老一辈的亲戚关系。现在春节除夕这种大日子也上班，越来越也无所谓节日团...
         546                              我真无语了，不行给隔壁申遗了吧[笑]
         547                                          强盗逻辑
         548                  老板大发慈悲就能回乡吃上年夜饭，老板不乐意你就得在异地点外卖给他干[爱你]
         549                             当算当保安了，爱谁建设谁建设，爱谁奉献谁奉献
         Name: text, Length: 550, dtype: object
```

```
In  [11]: df['text'] = df['text'].apply(emjio_tihuan)
          df['text']
```

```
Out[11]: 0                                四大发明：调休、公摊、996、灵活就业
         1                         一天到晚消费消费消费，就盯着老百姓口袋里的一点生活费
         2                      除夕回不了家＝欢乐祥和？？？你怕是不知道有多少人是背井离乡在外上班
         3                                         怎么不建议大家辞职呢
         4                                            消费个卵
                                        ...
         545    年轻人都去大城市，割断了老一辈的亲戚关系。现在春节除夕这种大日子也上班，越来越也无所谓节日团...
         546                                 我真无语了，不行给隔壁申遗了吧
         547                                          强盗逻辑
         548                      老板大发慈悲就能回乡吃上年夜饭，老板不乐意你就得在异地点外卖给他干
         549                             当算当保安了，爱谁建设谁建设，爱谁奉献谁奉献
         Name: text, Length: 550, dtype: object
```

```
In  [12]: df.dropna(subset=['text'], axis=0,inplace=True)
          df['text'] = df['text'].apply(yasuo)
          df['分词'] = df['text'].apply(get_cut_words)
          new_df = df.dropna(subset=['分词'], axis=0)
          new_df = new_df.drop_duplicates(subset=['分词'])
          new_df['分词']
```

```
Building prefix dict from the default dictionary ...
Loading model from cache C:\Users\V_YUHA~1\AppData\Local\Temp\jieba.cache
Loading model cost 0.642 seconds.
Prefix dict has been built successfully.
```

```
Out[12]: 0                                   四大发明 调休 公摊 灵活 就业
         1                             一天到晚 消费 老百姓 口袋 一点 生活费
         2                                除夕 欢乐祥和 背井离乡 在外 上班
         3                                            建议 辞职
         4                                              消费
                                        ...
         545    年轻人 大城市 割断 老一辈 亲戚关系 春节 除夕 日子 上班 越来越 无所谓 节日 团聚 ...
         546                                    真无语 不行 隔壁 申遗
         547                                           强盗 逻辑
         548                          老板 大发慈悲 回乡 年夜饭 老板 不乐意 地点 外卖
         549                                    当算 保安 建设 奉献
         Name: 分词, Length: 484, dtype: object
```

3、进行情感打标为接下来模型做情感分类做铺垫

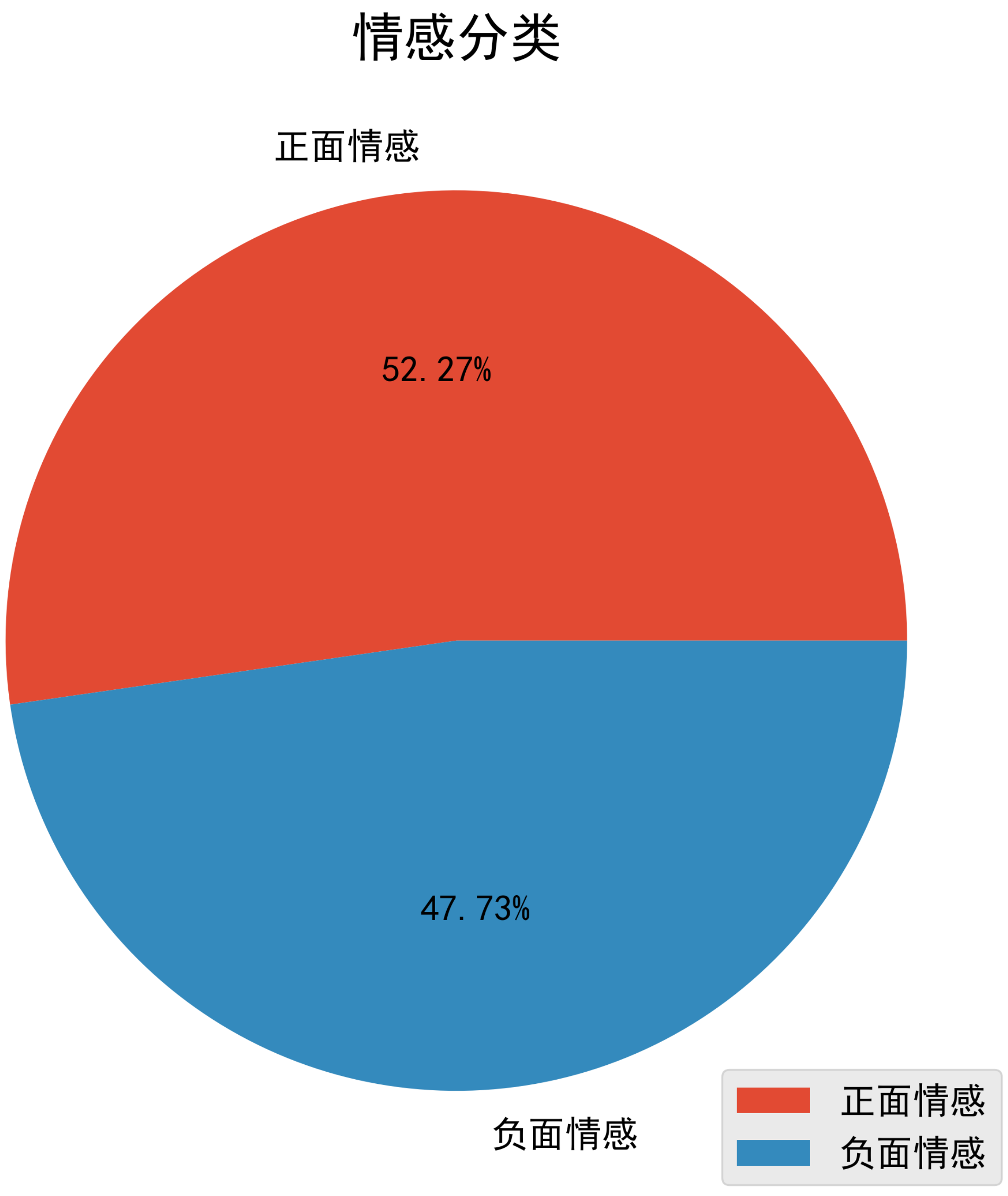
```
In  [13]: def analyze_sentiment(text):
              s = SnowNLP(text)
              sentiment = s.sentiments
              if sentiment >= 0.5:
                  return '正面情感'
              else:
                  return '负面情感'
```

```
In  [14]: new_df['情感分类'] = new_df['分词'].apply(analyze_sentiment)
          new_df['情感分类']
```

```
Out[14]: 0      正面情感
         1      正面情感
         2      正面情感
         3      负面情感
         4      负面情感
                ...
         545    正面情感
         546    负面情感
         547    正面情感
         548    正面情感
         549    负面情感
         Name: 情感分类, Length: 484, dtype: object
```

```
In  [15]: import matplotlib
          import matplotlib.pyplot as plt
          import seaborn as sns
```

```
In [16]: plt.style.use('ggplot')
         plt.rcParams['font.sans-serif'] = ['SimHei']
         plt.figure(dpi=500)
         new_df1 = new_df['情感分类'].value_counts()
         x_data = [str(x) for x in new_df1.index]
         y_data = [int(x) for x in new_df1.values]

         plt.pie(y_data, labels=x_data, startangle=0, autopct='%1.2f%%')
         plt.title('情感分类')
         # 添加图例
         plt.legend(x_data, loc='lower right')
         plt.tight_layout()
         plt.savefig('Sentiment classification.png')
```



In [17]:
```
!pip install stylecloud -i  https://pypi.douban.com/simple
!pip install IPython -i  https://pypi.douban.com/simple
```

```
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\resolvelib\resolvers.py", line 148, in _add_to_criteria
    matches = self._p.find_matches(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\provider.py", line 231, in find_matches
    return self._factory.find_candidates(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\factory.py", line 424, in find_candidate
s
    return self._iter_found_candidates(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\factory.py", line 320, in _iter_found_ca
ndidates
    _get_installed_candidate(),
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\factory.py", line 268, in _get_installed
_candidate
    candidate = self._make_candidate_from_dist(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\factory.py", line 160, in _make_candidat
e_from_dist
    base = AlreadyInstalledCandidate(dist, template, factory=self)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\candidates.py", line 351, in __init__
    factory.preparer.prepare_installed_requirement(self._ireq, skip_reason)

  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\operations\prepare.py", line 732, in prepare_installed_require
```

In [18]:
```
import stylecloud
from IPython.display import Image
```

```
Call stack:
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\runpy.py", line 197, in _run_module_as_main
    return _run_code(code, main_globals, None,
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\runpy.py", line 87, in _run_code
    exec(code, run_globals)
  File "D:\Users\v_yuhaozeng\Anaconda3\Scripts\pip.exe\__main__.py", line 7, in <module>
    sys.exit(main())
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\main.py", line 79, in main
    return command.main(cmd_args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\base_command.py", line 101, in main
    return self._main(args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\base_command.py", line 234, in _main
    return run(options, args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\base_command.py", line 180, in exc_logging_wrapper
    status = run_func(*args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\req_command.py", line 248, in wrapper
    return func(self, options, args)
```

In [19]:
```
d = {}
list_text = []
for t in df['分词']:
    # 把数据分开
    t = str(t).split(" ")
    for i in t:
        # 添加到列表里面
        list_text.append(i)
        d[i] = d.get(i,0)+1

ls = list(d.items())
ls.sort(key=lambda x:x[1],reverse=True)
x_data = []
y_data = []
for key,values in ls[:100]:
    x_data.append(key)
    y_data.append(values)

data = pd.DataFrame()
data['word'] = x_data
data['counts'] = y_data

data.to_csv('高频词Top100.csv',encoding='utf-8-sig',index=False)
# 然后传入词云图中，筛选最多的100个词
stylecloud.gen_stylecloud(text=' '.join(list_text), max_words=100,
                              # 不能有重复词
                              collocations=False,
                              max_font_size=400,
                              # 字体样式
                              font_path='simhei.ttf',
                              # 图片形状
                              icon_name='fas fa-circle',
                              # 图片大小
                              size=1200,
                              # palette='matplotlib.Inferno_9',
                              # 输出图片的名称和位置
                              output_name='词云图.png')
# 开始生成图片
Image(filename='词云图.png')
```

```
    return self._main(args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\base_command.py", line 234, in _main
    return run(options, args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\base_command.py", line 180, in exc_logging_wrapper
    status = run_func(*args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\cli\req_command.py", line 248, in wrapper
    return func(self, options, args)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\commands\install.py", line 377, in run
    requirement_set = resolver.resolve(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_internal\resolution\resolvelib\resolver.py", line 92, in resolve
    result = self._result = resolver.resolve(
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\resolvelib\resolvers.py", line 546, in resolve
    state = resolution.resolve(requirements, max_rounds=max_rounds)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\resolvelib\resolvers.py", line 427, in resolve
    failure_causes = self._attempt_to_pin_criterion(name)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\resolvelib\resolvers.py", line 239, in _attempt_to_pin_criterion
    criteria = self._get_updated_criteria(candidate)
  File "D:\Users\v_yuhaozeng\Anaconda3\lib\site-packages\pip\_vendor\resolvelib\resolvers.py", line 230, in _get_updated_criteria
    self._add_to_criteria(criteria, requirement, parent=candidate)
```

In [20]:
```
#填充空值
df['location'].fillna('其他',inplace=True)
```

In [21]:
```
#统计IP属地的空值
df['location'].isnull().value_counts()
```

Out[21]:
```
location
False    550
Name: count, dtype: int64
```

In [22]:
```
#检查重复
df.duplicated().value_counts()
```

Out[22]:
```
False    550
Name: count, dtype: int64
```

In [24]:
```
location=df['location']
location
```

Out[24]:
```
0          其他
1          重庆
2          其他
3      北京 海淀区
4       四川 成都
        ...
545        北京
546        其他
547     吉林 长春
548        其他
549        其他
Name: location, Length: 550, dtype: object
```

In [26]:
```
loc = []
for x in location:
    x=x.split()[0]
    loc.append(x)
loc
```

Out[26]:
```
['其他',
 '重庆',
 '其他',
 '北京',
 '四川',
 '北京',
 '重庆',
 '天津',
 '广东',
 '其他',
 '其他',
 '贵州',
 '广东',
 '重庆',
 '北京',
 '江西',
 '海外',
 '四川',
 '北京',
```

In [27]:
```
x=list(set(loc))
x
```

Out[27]:
```
['江西',
 '天津',
 '广西',
 '山东',
 '江苏',
 '辽宁',
 '甘肃',
 '上海',
 '黑龙江',
 '重庆',
 '贵州',
 '海外',
 '湖北',
 '湖南',
 '陕西',
 '海南',
 '浙江',
 '四川',
 '安徽',
 '福建',
 '西藏',
 '其他',
 '山西',
 '北京',
 '云南',
 '河南',
 '内蒙古',
 '广东',
 '河北',
 '吉林',
 '新疆']
```

In [28]:
```
dict = {}
for key in loc:
    dict[key] = dict.get(key, 0) + 1
dict
```

Out[28]:
```
{'其他': 176,
 '重庆': 11,
 '北京': 49,
 '四川': 34,
 '天津': 11,
 '广东': 37,
 '贵州': 8,
 '江西': 4,
 '海外': 16,
 '福建': 6,
 '安徽': 12,
 '浙江': 19,
 '山西': 7,
 '上海': 33,
 '吉林': 9,
 '海南': 6,
 '新疆': 3,
 '陕西': 8,
 '云南': 5,
 '河南': 10,
 '山东': 18,
 '湖北': 7,
 '西藏': 3,
 '江苏': 15,
 '河北': 10,
 '内蒙古': 5,
 '湖南': 4,
 '广西': 7,
 '辽宁': 5,
 '黑龙江': 10,
 '甘肃': 2}
```

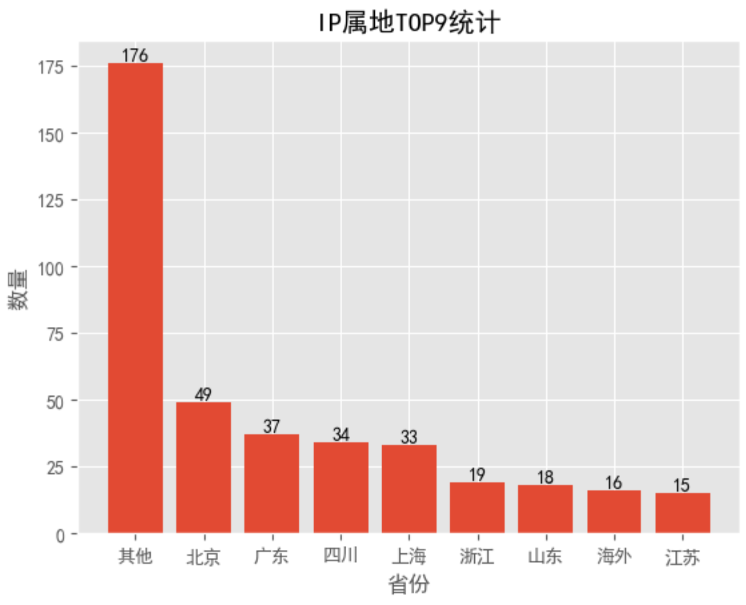
In [29]:
```
dict.items()
L=list(dict.items())       # 得到列表L
L.sort(key=lambda x:x[1],reverse=True)  # 按列表中，每一个元组的第二个元素从小到大排序,x代表从L中遍历出的一个元组
L
```

Out[29]:
```
[('其他', 176),
 ('北京', 49),
 ('广东', 37),
 ('四川', 34),
 ('上海', 33),
 ('浙江', 19),
 ('山东', 18),
 ('海外', 16),
 ('江苏', 15),
 ('安徽', 12),
 ('重庆', 11),
 ('天津', 11),
 ('河南', 10),
 ('河北', 10),
 ('黑龙江', 10),
 ('吉林', 9),
 ('贵州', 8),
 ('陕西', 8),
 ('山西', 7),
 ('湖北', 7),
 ('广西', 7),
 ('福建', 6),
 ('海南', 6),
 ('云南', 5),
 ('内蒙古', 5),
 ('辽宁', 5),
 ('江西', 4),
 ('湖南', 4),
 ('新疆', 3),
 ('西藏', 3),
 ('甘肃', 2)]
```

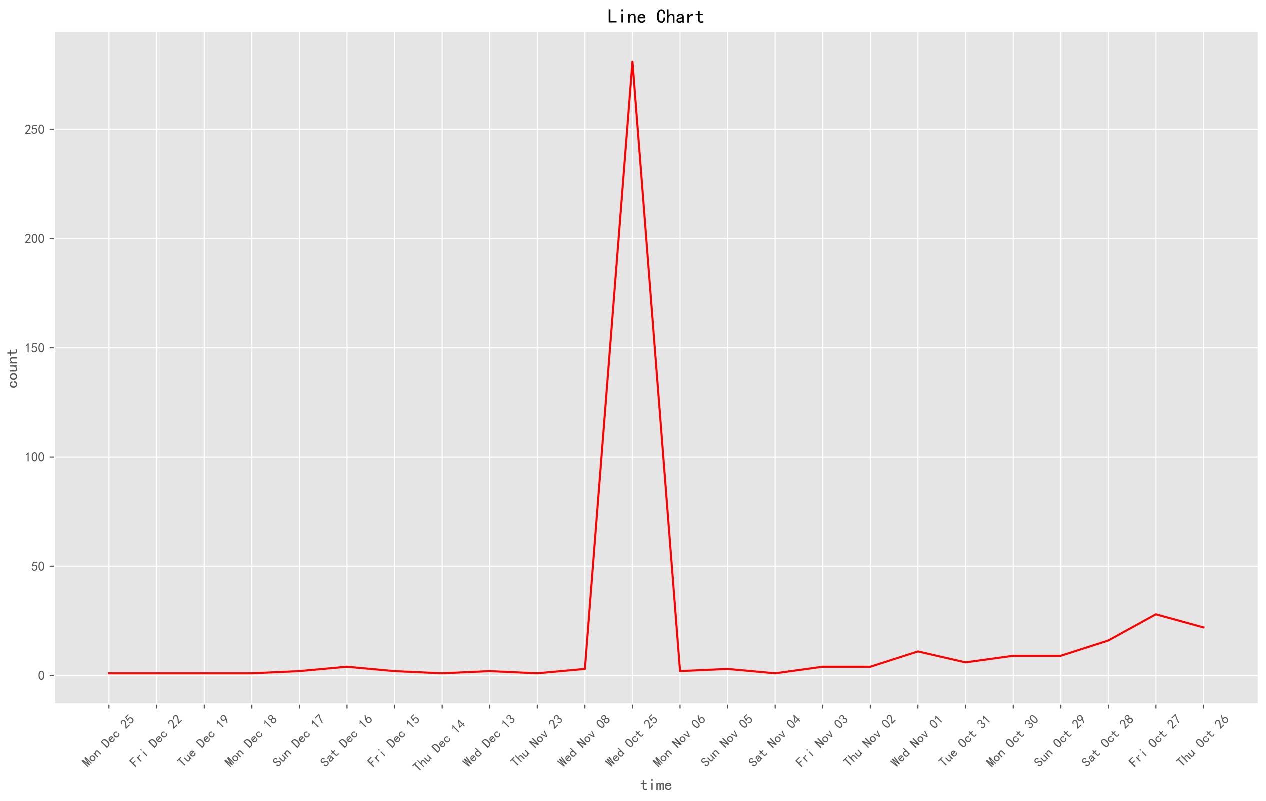
In [30]:
```
L=L[0:9]
dict_lst = {tup[0]: tup[1] for tup in L}
dict_lst
```

Out[30]:
```
{'其他': 176,
 '北京': 49,
 '广东': 37,
 '四川': 34,
 '上海': 33,
 '浙江': 19,
 '山东': 18,
 '海外': 16,
 '江苏': 15}
```

In [35]:
```
# 解决无法显示中文的问题
plt.rcParams["font.sans-serif"] = ["SimHei"]  # 设置字体
#dict_list
# 从大到小排序
y2 = {k: v for k, v in sorted(dict_lst.items(), key=lambda item: item[1], reverse=True)}
# 画柱状图
bars = plt.bar(dict_lst.keys(), height=dict_lst.values())

# 在柱子上添加数值
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, round(yval, 2), ha='center', va='bottom') # 通过调整0.005可以调整数字标签的位置
# 设置标题
plt.title('IP属地TOP9统计')
# 设置x轴名称
plt.xlabel('省份')
# 设置y轴名称
plt.ylabel('数量')
plt.savefig('IP属地TOP9统计.png')
plt.show()
```



In [39]:
```
# 读取Excel文件
df1 = pd.read_excel('时间数量.xlsx')
# 提取需要绘制的列
x = df1['time']
y = df1['number']
plt.figure(figsize=(16, 9),dpi=500)
# 绘制折线图
plt.plot(x, y,color='r')
# 设置图表标题和坐标轴标签
# 设置x轴的标签倾斜45度
plt.xticks(rotation=45)
plt.xlabel("time")
plt.ylabel("count")
plt.title("Line Chart")
plt.savefig('Line Chart.png')
plt.show()
```



In [39]:
```
from sklearn import svm
from sklearn.metrics import roc_curve, auc
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from scipy import interp
from sklearn.metrics import roc_auc_score
from sklearn.metrics import precision_score, recall_score, f1_score
```

In [26]:
```
def sentiment_type(x):
    x1 = str(x)
    if x1 == "正面情感":
        return 1
    else:
        return 0

new_df['情感分类'] = new_df['情感分类'].apply(sentiment_type)
new_df['情感分类']
```

Out[26]:
```
0      1
1      1
2      1
3      0
4      0
      ..
545    1
546    0
547    1
548    1
549    0
Name: 情感分类, Length: 484, dtype: int64
```

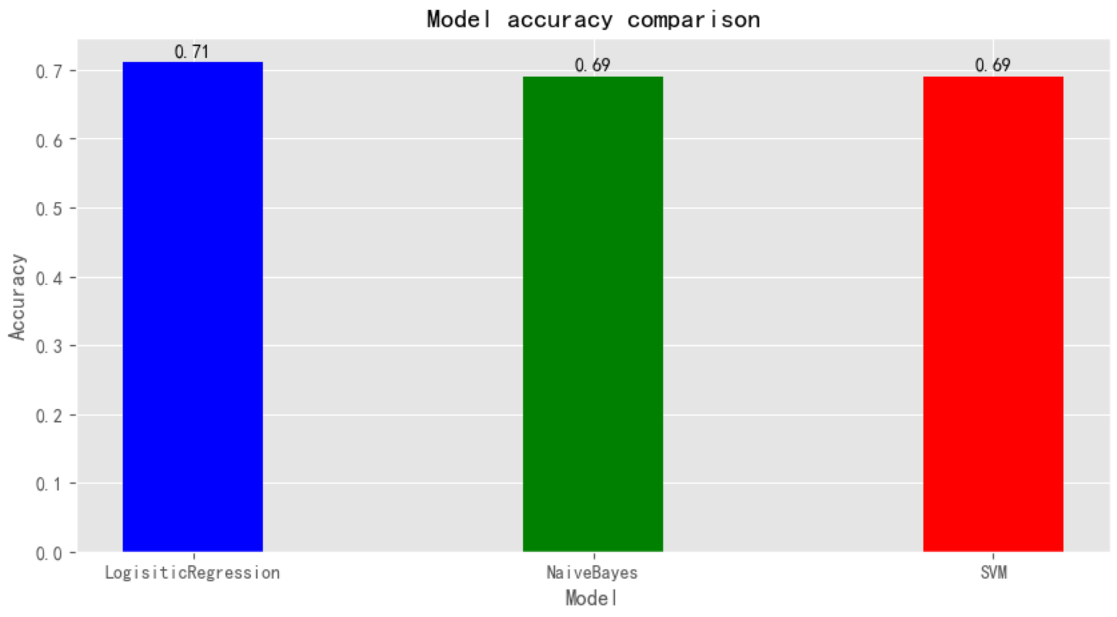
4、划分数据集，分割训练集和测试集、并且通过管道进行多模型训练，采用TF-IDF算法做特征向量，这里的模型采用了逻辑回归、贝叶斯和支持向量机

In [34]:
```
content = new_df['分词']
labels = new_df['情感分类']

# 通过TfidfVectorizer，将文本转换为特征向量
vectorizer = TfidfVectorizer()
content = vectorizer.fit_transform(content)

# 将数据集分割为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(content, labels, test_size=0.2, random_state=0)

# 训练模型
classifiers = {
    "LogisiticRegression": LogisticRegression(),
    "NaiveBayes": MultinomialNB(),
    "SVM": svm.SVC(probability=True)
}

accuracy_score_results = {}
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    accuracy = accuracy_score(y_test, predictions)
    accuracy_score_results[key] = accuracy

accuracy_score_results
```

Out[34]:
```
{'LogisiticRegression': 0.711340206185567,
 'NaiveBayes': 0.6907216494845361,
 'SVM': 0.6907216494845361}
```

5、获取每个模型的准确率数值并且用柱状图进行可视化

In [47]:
```
x_data = list(accuracy_score_results.keys())
y_data = list(accuracy_score_results.values())
# 绘制每种模型的准确度比较柱状图
plt.figure(figsize=(10,5))
bars = plt.bar(x_data, y_data, color=['b', 'g', 'r'],width=0.35)
plt.title('Model accuracy comparison')
plt.xlabel('Model')
plt.ylabel('Accuracy')


# 在柱子上添加数值
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, round(yval, 2), ha='center', va='bottom') # 通过调整0.005可以调整数字标签的位置
plt.savefig('Model accuracy comparison.png')
plt.show()
```

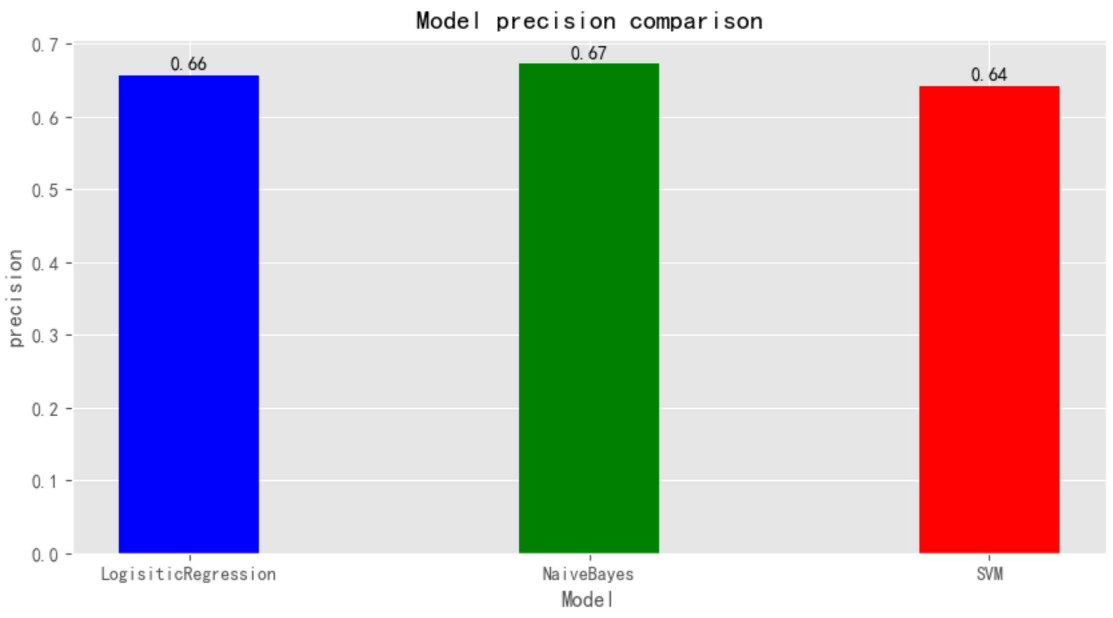


In [40]:
```
precision_score_results = {}
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    precision = precision_score(y_test, predictions)
    precision_score_results[key] = precision

precision_score_results
```

Out[40]:
```
{'LogisiticRegression': 0.6567164179104478,
 'NaiveBayes': 0.6727272727272727,
 'SVM': 0.6417910447761194}
```

6、获取每个模型的精确率数值并且用柱状图进行可视化

In [48]:
```
x_data = list(precision_score_results.keys())
y_data = list(precision_score_results.values())
# 绘制每种模型的准确度比较柱状图
plt.figure(figsize=(10,5))
bars = plt.bar(x_data, y_data, color=['b', 'g', 'r'],width=0.35)
plt.title('Model precision comparison')
plt.xlabel('Model')
plt.ylabel('precision')


# 在柱子上添加数值
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, round(yval, 2), ha='center', va='bottom') # 通过调整0.005可以调整数字标签的位置
plt.savefig('Model precision comparison.png')
plt.show()
```

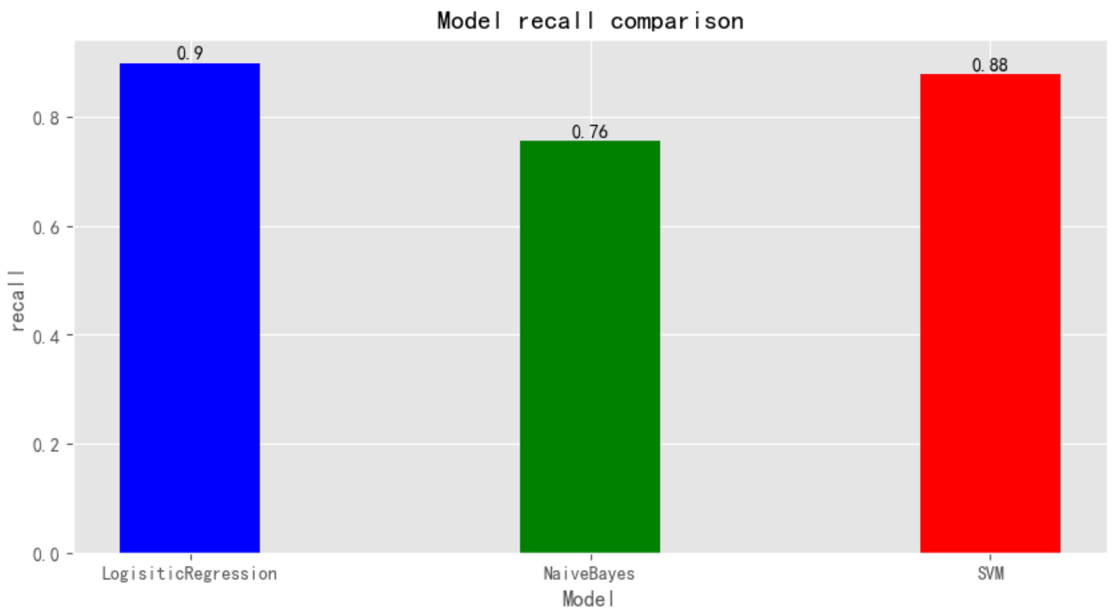


In [42]:
```
recall_score_results = {}
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    recall = recall_score(y_test, predictions)
    recall_score_results[key] = recall

recall_score_results
```

Out[42]:
```
{'LogisiticRegression': 0.8979591836734694,
 'NaiveBayes': 0.7551020408163265,
 'SVM': 0.8775510204081632}
```

7、获取每个模型的召回率数值并且用柱状图进行可视化

In [49]:
```
x_data = list(recall_score_results.keys())
y_data = list(recall_score_results.values())
# 绘制每种模型的准确度比较柱状图
plt.figure(figsize=(10,5))
bars = plt.bar(x_data, y_data, color=['b', 'g', 'r'],width=0.35)
plt.title('Model recall comparison')
plt.xlabel('Model')
plt.ylabel('recall')


# 在柱子上添加数值
for bar in bars:
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, round(yval, 2), ha='center', va='bottom') # 通过调整0.005可以调整数字标签的位置
plt.savefig('Model recall comparison.png')
plt.show()
```

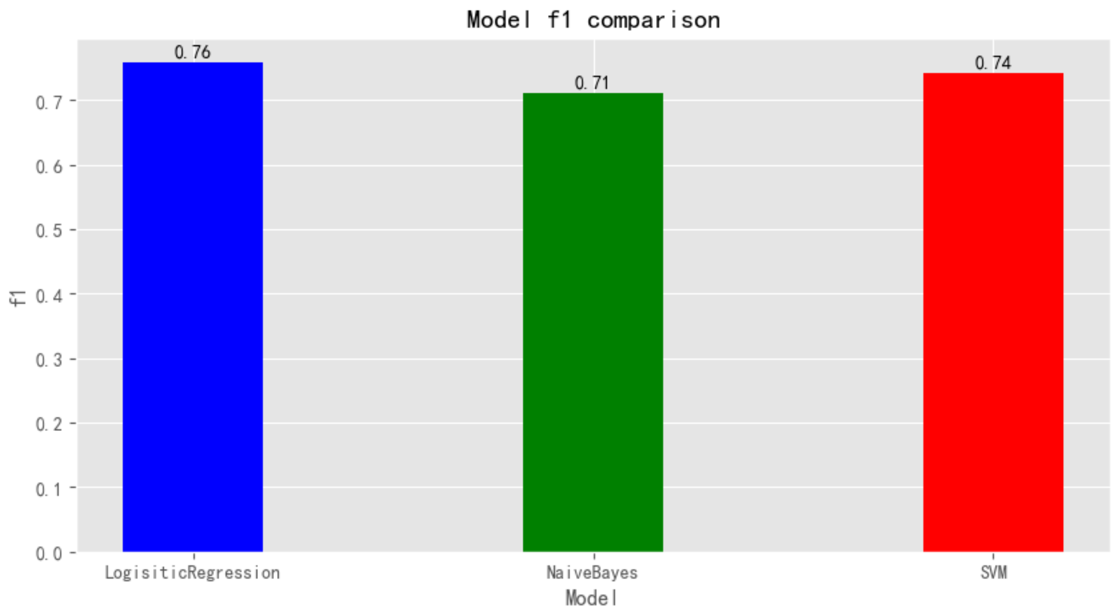


In [45]:
```
f1_score_results = {}
for key, classifier in classifiers.items():
    classifier.fit(X_train, y_train)
    predictions = classifier.predict(X_test)
    f1 = f1_score(y_test, predictions)
    f1_score_results[key] = f1

f1_score_results
```

Out[45]:
```
{'LogisiticRegression': 0.7586206896551725,
 'NaiveBayes': 0.7115384615384616,
 'SVM': 0.7413793103448276}
```

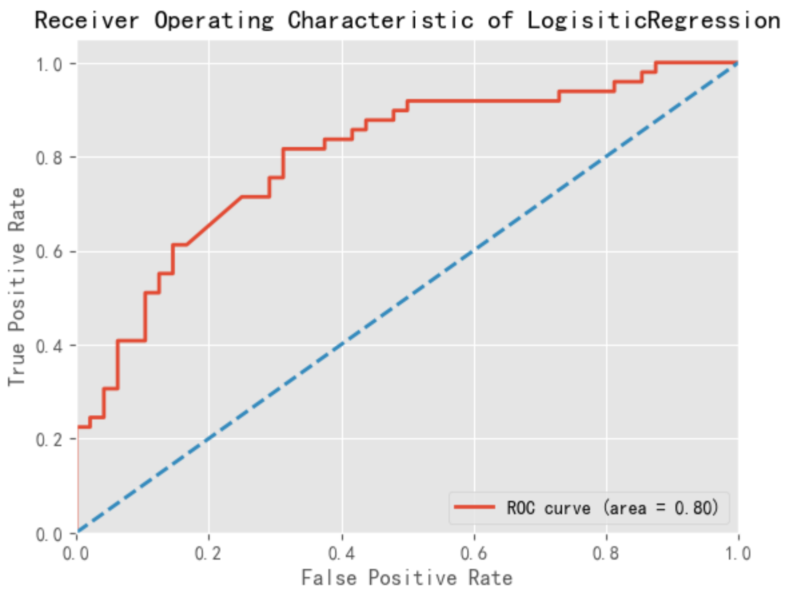
8、获取每个模型的F1数值并且用柱状图进行可视化

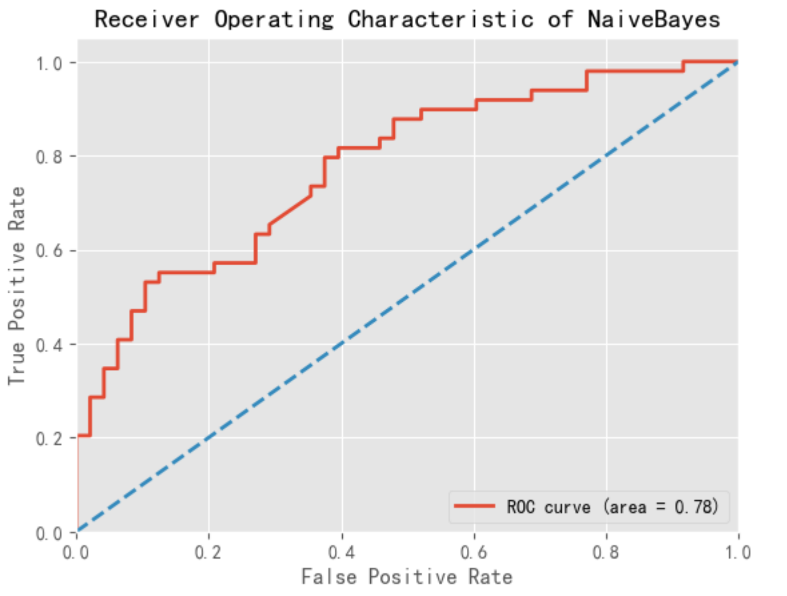
```
In  [50]: x_data = list(f1_score_results.keys())
          y_data = list(f1_score_results.values())
          # 绘制每种模型的准确度比较柱状图
          plt.figure(figsize=(10,5))
          bars = plt.bar(x_data, y_data, color=['b', 'g', 'r'],width=0.35)
          plt.title('Model f1 comparison')
          plt.xlabel('Model')
          plt.ylabel('f1')


          # 在柱子上添加数值
          for bar in bars:
              yval = bar.get_height()
              plt.text(bar.get_x() + bar.get_width()/2, yval + 0.005, round(yval, 2), ha='center', va='bottom') # 通过调整0.005可以调整数字标签的位置
          plt.savefig('Model f1 comparison.png')
          plt.show()
```

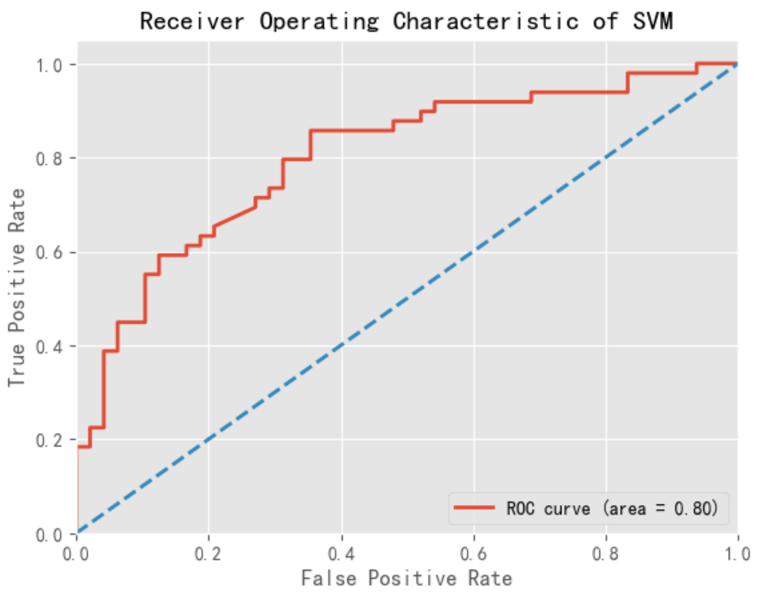


9、获取每个模型的ROC值并且用曲线图进行可视化

In [53]:
```
# 对各个分类器的预测结果进行评估
from sklearn import preprocessing
for (name, classifier) in classifiers.items():
    classifier.fit(X_train, y_train)
    y_score = classifier.predict_proba(X_test)

    # 计算roc和auc
    fpr, tpr, thresholds = roc_curve(y_test, y_score[:, 1])
    roc_auc = auc(fpr, tpr)

    # 绘制ROC曲线
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic of '+ name)
    plt.legend(loc="lower right")
    plt.savefig('Receiver Operating Characteristic of '+ name + '.png')
    plt.show()
```






Receiver Operating Characteristic of SVM
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.80)