一、消费数据分析

1、 介绍：分析一全球跨国企业在各地的产品消费情况

2、 数据：采用 kaggle 竞赛网上提供的数据消费记录，纪录主要包括发票号码，产品编号，消费的产品数量，时间，产品单价，客户国家等

3、 工具和方法：学习利用 python 进行简单的数据分析操作

第一步：首先对原始数据进行清洗

	A	B	C	D	E	F	G	H
1	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
2	536365	85123A	WHITE HANGING HEART T-LIGHT	6	2010/12/1 8:26	2.55	17850	United Kingdom
3	536365	71053	WHITE METAL LANTERN	6	2010/12/1 8:26	3.39	17850	United Kingdom
4	536365	84406B	CREAM CUPID HEARTS COAT HA	8	2010/12/1 8:26	2.75	17850	United Kingdom
5	536365	84029G	KNITTED UNION FLAG HOT WAT	6	2010/12/1 8:26	3.39	17850	United Kingdom
6	536365	84029E	RED WOOLLY HOTTIE WHITE HE	6	2010/12/1 8:26	3.39	17850	United Kingdom
7	536365	22752	SET 7 BABUSHKA NESTING BOX	2	2010/12/1 8:26	7.65	17850	United Kingdom
8	536365	21730	GLASS STAR FROSTED T-LIGHT	6	2010/12/1 8:26	4.25	17850	United Kingdom
9	536366	22633	HAND WARMER UNION JACK	6	2010/12/1 8:28	1.85	17850	United Kingdom

由于原始数据量在 50 万左右，里面存在部分空值和重复值，数据量比较大，所以可以直接把这些数据剔除掉

```
def clean_data(datafile):
    adatafile = datafile.dropna()
    print(datafile.shape)
    print(adatafile.shape)
    bdatadile = adatafile.drop_duplicates()
    print(bdatadile.shape)
    bdatadile.to_csv(data_out,index=False,encoding='utf-8')
    return bdatadile
```

剔除前后数据量可查看

```
C:\Users\JIWei\Anaconda3\pyt
(541909, 8)
(406829, 8)
(401604, 8)
```

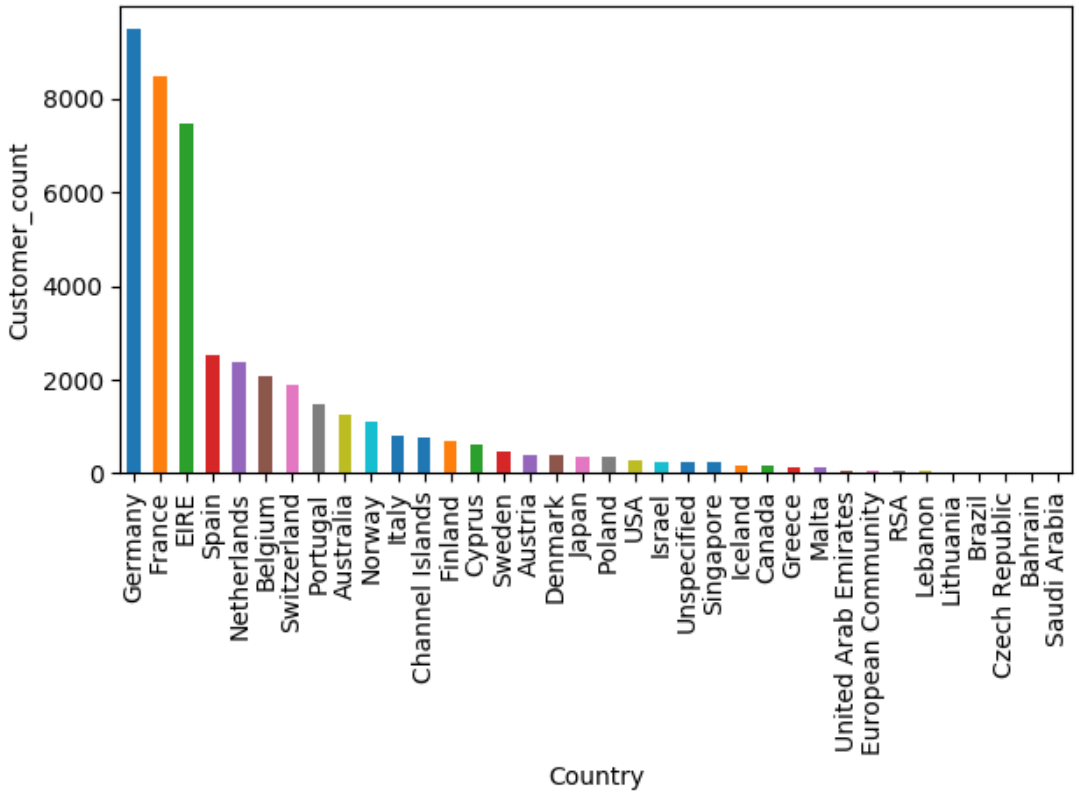
剔除后仍然有 40 万的数据用来分析

第二步：进行一些简要的数据分析

首先分析比较各国家的客户数量，看看该企业在哪些国家有较大的客户群体

```
def show_pic(datafile):
    customer_per_country = datafile['Country'].value_counts()

    print(customer_per_country)
    #print(customer_per_country_df)
    #customer_per_country
    #sns.barplot(customer_per_country_df)
    customer_per_country_count = customer_per_country[customer_per_country.index != 'United Kingdom']
    print(customer_per_country_count)
    customer_per_country_count.plot(kind='bar')
    #plt.xticks(rotation = 90)
    plt.xlabel('Country')
    plt.ylabel('Customer_count')
    plt.tight_layout()
```

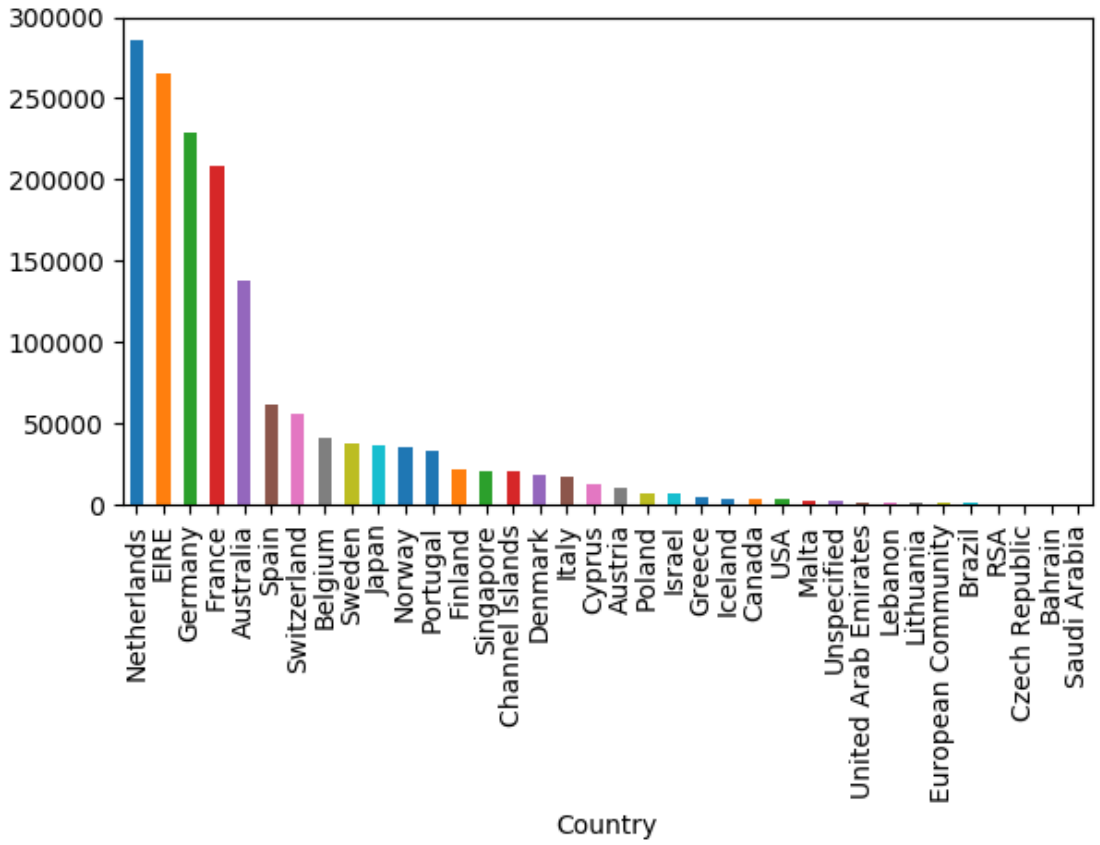


由于英国的客户数量要远远大于其他国家，所以这里分析是把英国的客户数据给排除了，也可以推测出该公司可能就是英国起家，总部在英国，所以对本国市场比较熟悉，具有较大客户群体。其次，除英国外，客户数量比较大的是德国、法国和爱尔兰，这几个国家和英国同属发达国家，互相经济关系联系也比较紧密，市场打通也不错，客户群体较高。其他一些客户数量较大的也都是一些欧洲邻国。

可以得出结论：该公司产品在和英国相邻的几个发达国家有比较大的客户群体，可能是因为彼此的文化、生活等差别不大，产品使用率高，相反其他地方的客户群体较少，可以尝试根据当地文化和生活对产品进行针对性的改良，获取客户群体市场。

其次分析比较各国家的成交额，看看哪些国家成交额数量较大

```
def show_pic2(datadile):
    data1 = datadile['Country'] != 'United Kingdom'
    data2 = ~datadile['InvoiceNo'].str.startswith('C')
    valid_data = datadile[data1 & data2].copy()
    valid_data['total_cost'] = valid_data['Quantity'] * valid_data['UnitPrice']
    cost_per_country = valid_data.groupby('Country')['total_cost'].sum()
    print(cost_per_country)
    cost_per_country.sort_values(ascending=False).plot(kind='bar')
    plt.tight_layout()
    plt.savefig('./output/show_pic2.png')
```



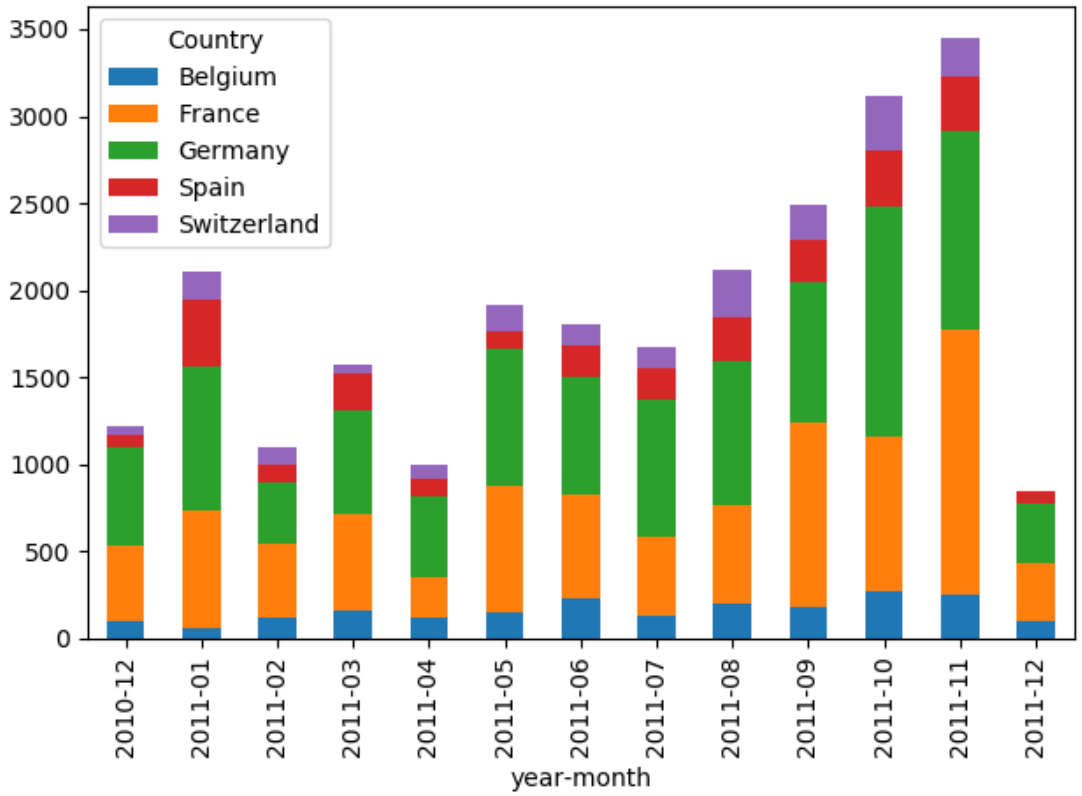
可以看出成交额前五名是荷兰、爱尔兰、德国、法国和意大利，爱尔兰、德国和法国因为之前分析过客户群体较大，所以也应当有较高的成交额，那么荷兰和意大利较高的成交额可能是因为有大量的集体批发商，他们客户群体的数量不多，但是一次性批发的产品数量较大。其次一些像日本和新加坡等亚洲国家的成交额也和一些欧洲国家诸如意大利和丹麦等成交额相当，说明该产品在除了欧洲外，也有一定的市场，但应该是以批发客户为主。

总的来看，该产品在不同国家对应的客户群体不一样，有些以零售客户为主，有些以批发客户为主，该公司可以在不同国家针对不同的群体制定相应的销售措施，以获取最大利润。

最后再来看看各国家的交易记录趋势，随机选取德国、法国、西班牙、比利时和瑞士这五个国家来分析随时间的产品交易趋势。

```
def show_pic3(datafile):
    country = ['Germany', 'France', 'Spain', 'Belgium', 'Switzerland']
    valid_data = datafile[datafile['Country'].isin(country)].copy()
    #print(valid_data.info())
    #print(valid_data.head())
    valid_data['InvoiceDate'] = pd.to_datetime(valid_data['InvoiceDate'])
    valid_data['year'] = valid_data['InvoiceDate'].dt.year.astype(str)
    valid_data['month'] = valid_data['InvoiceDate'].dt.month.astype(str)
    valid_data['year-month'] = valid_data['year'].str.cat(valid_data['month'],sep='-')
    #print(valid_data.head())
    month_country_count = valid_data.groupby(['year-month', 'Country'])['StockCode'].count()
    #print(month_country_count)

    month_country_count_pf = month_country_count.unstack()
    month_country_count_pf.index = pd.to_datetime(month_country_count_pf.index).to_period('M')
    month_country_count_pf.sort_index(inplace=True)
    print(month_country_count_pf)
    month_country_count_pf.plot(kind='bar',stacked=True)
```



从图中可以简单判断出不同的月份该产品在市场的交易量还是不同的，11 年整体趋势是上升的，但是到 12 月份有一个急剧的下降，因为没有后期数据，不好判断后期走势，所以也不好判断是市场因素还是时间因素，不过可以借鉴的是根据历史数据进行之后每个月的生产数量的简单判断，以做到这个生产数量和库存的适当配比，以免过多的生产浪费产生过多库存或市场供需不足。

二、违约贷款预测分析

1、 介绍：根据贷款记录各项信息预测该笔贷款是否有风险

2、 数据：利用数据竞赛网 kaggle 上提供的贷款数据，数据信息包括年龄、性别、工作、住房、账户、违约情况等信息

3、 方法：利用 python 和 sklearn 机器学习库进行分析（kNN 和 LR 分类方法）

第一步：查看元数据信息

	A	B	C	D	E	F	G	H	I	J
1	Age	Sex	Job	Housing	Saving ac	Checking	Credit am	Duration	Purpose	Risk
2	67	male	2	own	NA	little	1169	6	radio/TV	good
3	22	female	2	own	little	moderate	5951	48	radio/TV	bad
4	49	male	1	own	little	NA	2096	12	educatior	good
5	45	male	2	free	little	little	7882	42	furniture	good
6	53	male	2	free	little	little	4870	24	car	bad
7	35	male	1	free	NA	NA	9055	36	educatior	good
8	53	male	2	own	quite ric	NA	2835	24	furniture	good
9	35	male	3	rent	little	moderate	6948	36	car	good
10	61	male	1	own	rich	NA	3059	12	radio/TV	good

从原始数据可以发现，数据包含整形和字符串，还有部分缺失值，为了分类计算，需要把其他类别数据转为整形，同时需要对 NA 缺失数据处理，优于数据量整体只有 1000，所以没有剔除缺失值，且只有账户类型有缺失，所以用中值（moderate）取代，数据预处理如下：

首先构建数据映射字典

```
feature = ['Age','Sex','Job','Housing','Saving accounts','Checking account','Credit amount','Duration','Purpose']
label = ['Risk']

sex = {'male': 0, 'female': 1}
housing = {'free': 0, 'rent': 1, 'own': 2}
saving = {'little': 0, 'moderate': 1, 'rich': 2, 'quite rich': 3}
checking = {'little': 0, 'moderate': 1, 'rich': 2}
purpose = \
    {'radio/TV': 0,
     'education': 1,
     'furniture/equipment': 2,
     'car' : 3
```

然后对数据进行清洗和转换

```
def data_clean(data_in):
    data_out = data_in.fillna('moderate')
    return data_out

def data_trans(data_in):
    data_out = data_in.copy()
    data_out['Sex'] = data_in['Sex'].map(data_map.sex)
    data_out['Housing'] = data_in['Housing'].map(data_map.housing)
    data_out['Saving accounts'] = data_in['Saving accounts'].map(data_map.saving)
    data_out['Checking account'] = data_in['Checking account'].map(data_map.checking)
    data_out['Purpose'] = data_in['Purpose'].map(data_map.purpose)
    data_out['Risk'] = data_in['Risk'].map(data_map.risk)
```

可以查看清洗和转换后的数据如下：

	A	B	C	D	E	F	G	H	I	J
	Age	Sex	Job	Housing	Saving ac	Checking	Credit am	Duration	Purpose	Risk
	67	male	2	own	moderate	little	1169	6	radio/TV	good
	22	female	2	own	little	moderate	5951	48	radio/TV	bad
	49	male	1	own	little	moderate	2096	12	educatior	good
	45	male	2	free	little	little	7882	42	furniture	good
	53	male	2	free	little	little	4870	24	car	bad
	35	male	1	free	moderate	moderate	9055	36	educatior	good
	53	male	2	own	quite ric	moderate	2835	24	furniture	good
	35	male	3	rent	little	moderate	6948	36	car	good

特征：

```
[5 rows x 10 columns]
[[ 67   0   2 ... 1169   6   0]
 [ 22   1   2 ... 5951  48   0]
 [ 49   0   1 ... 2096  12   1]
 ...
 [ 38   0   2 ...  804  12   0]
 [ 23   0   2 ... 1845  45   0]
```

标签：

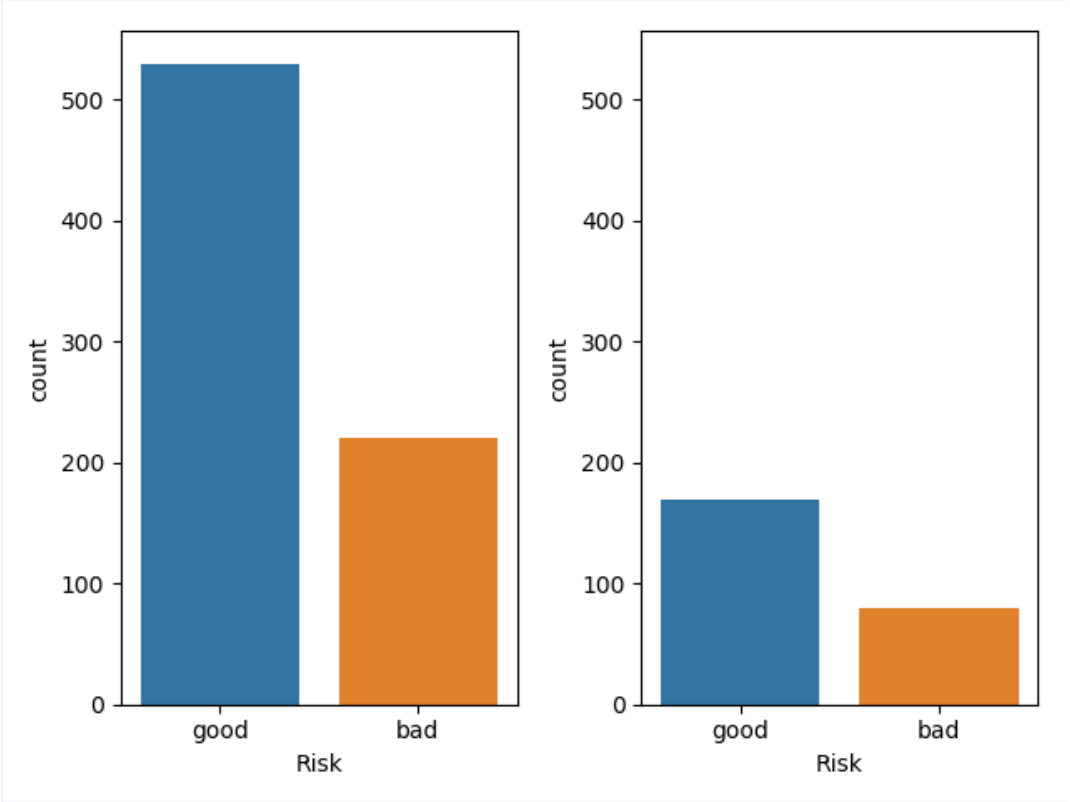
```
[[1]
 [0]
 [1]
 [1]
 [0]
```

第二步：对数据集进行划分，取 3/4 的数据进行训练，用剩下 1/4 数据进行测试

```
train_data, test_data = train_test_split(clean_data, test_size=1/4, random_state=10)
data_process.dataset_view(train_data,test_data)

X_train, y_train = data_process.data_trans(train_data)
X_test, y_test = data_process.data_trans(test_data)
model_name_para_dic = {'kNN':[5,10,15],
                       'LR':[0.01,1,100]}
```

顺便查看下训练数据和测试数据的分类情况



第三步：用 sklearn 库中 kNN 和 LR 两种方法进行分类比较，选取不同的参数，对比分类准确性和时间效率

```
def train_test_model(X_train, y_train, X_test, y_test, parametre, model_name):
    models = []
    accuracy = []
    times =[]
    for para in parametre:
        #print(para)
        if model_name == 'kNN':
            #print('训练kNN(k={})'.format(para))
            model = KNeighborsClassifier(n_neighbors=para)
        elif model_name == 'LR':
            #print('训练LR(c={})'.format(para))
            model = LogisticRegression(C=para)

        starttime = time.time()
        model.fit(X_train, y_train)
        endtime = time.time()
```

其中参数人为选取，kNN 选取 k 值参数分别为 5，10 和 15，LR 选取正则因子分别为 0.01，1 和 100，分别根据不同的参数训练和测试结果进行比较，得出最优模型和参数

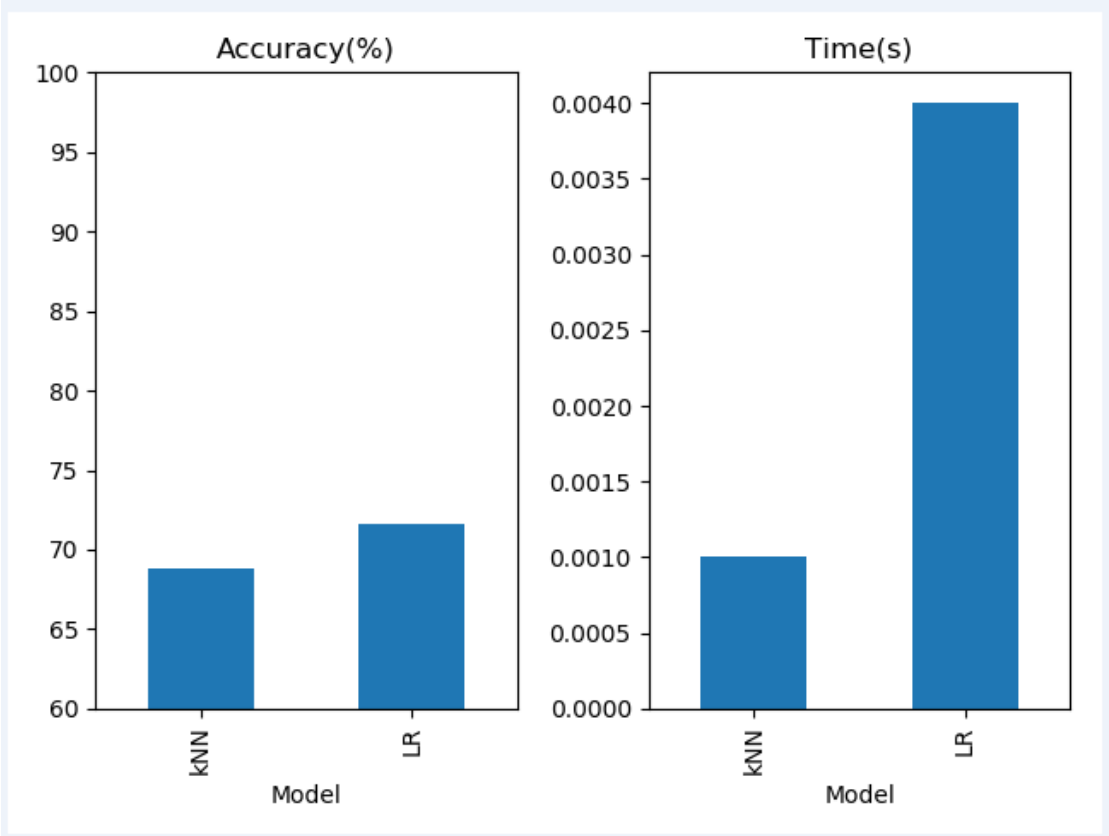
kNN

```
68.8
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=15, p=2,
           weights='uniform')
0.0010015169779459636
```

LR

```
71.6
LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
0.004002809524536133
```

分别将两种方法的准确率和时间效率用图形展现



可见，在当前数据集和选取参数情况下，LR 的分类准确性更高一点，但运行时间也更长，因为数据集不大，所以模型训练时间都在实际范围内，如果在大数据量比如 TB 甚至 PB 的量级，则需要综合考量准确性和模型训练效率。

通过练习能够让自己掌握整个机器学习和数据分析的流程，练习 python 代码的编写，学习使用一些简单的机器学习算法，也为数据分析的学习打基础。