目录

1 财务数据预处理

1.1 营收利润初步筛选

小组首先读取了企业营收利润数据表（LR.csv），使用 python 了 pandas 库进行数据的筛选，先把表 1 的需要字段分离，其次将“Typrep”字段值为“A”的数据收集并保存到 LR_1.csv 中，同时把目标编码设置为 UTF-8,最后使用 shape 函数来返回 LR_1.csv 整个数据矩阵的维度。

```
F: > import > 1_1.py > ...
1   import pandas as pd
2
3   lr = pd.read_csv("LR.csv")
4   selected_columns = ['Stkcd',    'Accper',   'Typrep',
5                       'B001000000',   'B001100000',   'B001101000',   'B001200000',
6                       'B001201000',   'B001207000',   'B001209000',   'B001210000',
7                       'B001211000',   'B001212000',   'B001303000',   'B002300000',]
8
9   lr_subset = lr[selected_columns]
10  lr_filtered = lr_subset[lr_subset['Typrep'] == 'A']
11  lr_filtered.to_csv("LR_1.csv", encoding='utf-8', index=False)
12  print("筛选后的数据行数和列数：", lr_filtered.shape)
13
```

任务 1.1 核心代码展示

在使用 python 导出矩阵维度的过程中，由于数据量较大，为了保证结果的正确，小组使用 Excel 中的 countif 函数直接计算了“Typrep”字段值为“A”的数据个数并和代码运行结果进行对比，两者完全相同，即**筛选后的行数为 33414，列数为 15。**

1.2 资产负债和对应行业合并

小组使用 python 的 pandas 库进行数据的处理，首先将“ZCFZ.csv”中的和“LR_1.csv”的“Stkcd”、“Accper”和“Typrep”三个字段进行匹配，随后将“ZCFZ.csv”对应的中字段为“A002000000”和“A001000000”的相应数据加入到“LR_1.csv”对应字段的末尾处。随后在“LR_1.csv”中的“Stkcd”和“Stk_ind.csv”中的同名字段进行匹配，把对应的“Indnme”和“Nindnme”的相应数据加入到“LR_1.csv”对应字段的末尾处，另存为“LR_2.csv”（UTF-8）。同时按照 1.1 的操作进行数据量的核对，确认无误。**最后所得行数为 33414，列数为 19。**

F: > import > 1_2.py > ...

```python
import pandas as pd
lr_data = pd.read_csv("LR_1.csv")
zcfz_data = pd.read_csv("ZCFZ.csv")
stk_ind_data = pd.read_csv("Stk_ind.csv", encoding='gbk')

#提取字段为“A002000000”和“A001000000”的数据
merged_zcfz = zcfz_data[(zcfz_data['Stkcd'].isin(lr_data['Stkcd'])) &
                        (zcfz_data['Accper'].isin(lr_data['Accper'])) &
                        (zcfz_data['Typrep'].isin(lr_data['Typrep']))]

#提取字段为“Indnme”和“Nindnme”的数据
merged_stk_ind = stk_ind_data[stk_ind_data['Stkcd'].isin(lr_data['Stkcd'])]

# 合并数据到LR_1.csv
lr_2_data = pd.merge(lr_data, merged_zcfz[['Stkcd', 'Accper', 'Typrep', 'A002000000', 'A001000000']],
                     on=['Stkcd', 'Accper', 'Typrep'], how='left')
lr_2_data = pd.merge(lr_2_data, merged_stk_ind[[
                     'Stkcd', 'Indnme', 'Nindnme']], on='Stkcd', how='left')

# 保存合并后的数据到LR_2.csv
lr_2_data.to_csv("LR_2.csv", encoding='gbk', index=False)

# 输出合并后数据的行数、列数
print(f"合并后数据的行数: {lr_2_data.shape[0]}, 列数: {lr_2_data.shape[1]}")
```

任务 1.2 的核心代码

1.3 删除无效数据

小组在“LR_2.csv”中查询值为 NULL 的数据，随后按列进行查找。当一列比对完成后发现该列空数据量小于 70%则放进特定的容器中，全部比对完毕后，对容器中的列数据再次进行计算，确认有效数据量小于 70%后移除，将剩下的数据移动至“LR_3.csv”（UTF-8）中。**处理后数据的列数为 17。**

同上方式，读取“LR_3.csv”后开始对行进行处理。和上文处理方式不同，行中出现空白数据即将一行全部删除，省去了验证步骤，简化代码，加快了程序进度。将未删除的数据进行确认完整后移动到“LR_4.csv”（UTF-8）中。**处理后数据的行数为 30888。**

F: > import > 1_3.py > ...

```python
import pandas as pd

lr_2 = pd.read_csv("LR_2.csv")
missing_percentage = lr_2.isnull().mean()
selected_columns = missing_percentage[missing_percentage < 0.7].index

lr_3 = lr_2[selected_columns]
lr_3.to_csv("LR_3.csv", encoding='utf-8', index=False)

print("处理后数据的列数：", lr_3.shape[1])

```

任务 1.3 的部分代码展示

F: > import > 1_4.py > ...

```python
import pandas as pd

lr_3 = pd.read_csv("LR_3.csv")
lr_4 = lr_3.dropna()
lr_4.to_csv("LR_4.csv", encoding='utf-8', index=False)

print("处理后数据的行数：", lr_4.shape[0])
```

任务 1.4 的部分代码展示

LR_4.csv 部分内容展示

1.4 日期转换

在“LR_4.csv”（UTF-8）中选取字段“Accper”的日期数据，将其拆分为Y/M/D的形式，随后转换为Y-M-D的形式，导出为LR_5.csv。具体实现方式如代码图所示。

```
F: > import > 1_5.py > ...
1   import pandas as pd
2
3   lr_4 = pd.read_csv("LR_4_new.csv")
4   lr_4['Accper'] = pd.to_datetime(
5       lr_4['Accper'], format='%Y/%m/%d').dt.strftime('%Y-%m-%d')
6
7   lr_4.to_csv("LR_5.csv", encoding='utf-8', index=False)
8
```

任务1.5代码展示

File Edit View

```
Stkcd,Accper,Typrep,B001000000,B001100000,B001101000,B001200000,B001201000,B001207000,B001209000,B001210000,B0012
600696,2018-03-31,A,8669264.35,18784589.8,18784589.8,10689923.22,7810739.35,204595.11,281510.68,2260219.43,115431
547,2018-03-31,A,96757726.45,517945246.5,517945246.5,426691627.5,272951740.1,789386.98,12657162.87,134498788.0,25
2772,2018-03-31,A,89533624.86,267161661.1,267161661.1,189823265.9,152798488.1,694050.24,30133399.88,9984416.73,-3
818,2018-03-31,A,174146344.7,907539547.9,907539547.9,733869064.7,653598292.4,15336943.77,24841913.58,34758925.04,
300568,2018-03-31,A,106998276.4,171703944.2,171703944.2,121289130.4,80665569.24,2874172.01,3784963.29,27099023.95
300282,2018-03-31,A,20772844.36,123728974.4,123728974.4,108734443.6,74363104.91,658046.92,7425442.63,21355510.54,
2313,2018-03-31,A,-4894755.2,597608156.6,597608156.6,607652329.9,481255016.5,2495854.73,35148084.11,72729449.87,1
600705,2018-03-31,A,1357163873.0,2779306478.0,1470703392.0,1692376287.0,825106311.6,15429945.69,344849928.2,25346
603637,2018-03-31,A,5929397.88,72483512.45,72483512.45,72829133.91,67715735.63,163533.06,697828.21,4929164.43,-18
603369,2018-03-31,A,676777504.5,1491542807.0,1490190804.0,852790917.8,382291886.4,208963349.2,216671128.4,4667419
2818,2018-03-31,A,209934751.2,328241076.4,328241076.4,123917329.2,92294338.38,19685611.98,367808.26,10704879.75,1
300114,2018-03-31,A,31821984.29,305112585.0,305112585.0,276732852.7,199184360.3,3778364.34,30231904.93,41060964.8
603788,2018-03-31,A,96316075.7,384411509.1,384411509.1,300964545.3,254558390.6,3521302.34,15810209.24,22945707.15
600961,2018-03-31,A,-30893815.17,2778633073.0,2778633073.0,2866389722.0,2744147158.0,7047663.47,16244726.5,440003
730,2018-03-31,A,83069250.05,1445922687.0,1445922687.0,1368517684.0,1001979693.0,13162135.26,162865667.8,15502434
```

LR_5.csv日期修改效果如图

1.5 计算利润率和资产负债率

读取LR_5.csv，新插入两列“利润率”和“资产负债率”，在python中使用pandas 库进行利润率和资产负债率的运算，通过题目中所给的公式利润总额（B001000000）/营业总收入（B001100000）和负债合计（A002000000）/资产总计（A001000000）得出了代码中的lr_5['利润率'] = (lr_5['B001000000'] / lr_5['B001100000'])、lr_5['资产负债率'] = (lr_5['A002000000'] / lr_5['A001000000'])部分。

添加字段后，将利润率、资产负债率小于 300%或者大于 300%的加入到特定的容器中，剩余的数据则保存在原数据表中。在对删除的数据进行二次确认后导出“LR_new.csv”（UTF-8），通过 shape 函数导出数据行数和列数。在 excel 表格中查询到前 5 个企业的数据。**处理后的数据行数和列数分别为 2073 和 36，**前 5 个企业的数据如下表格所示。

	利润率	资产负债率
1	0.461509	0.595244
2	0.186811	0.225886
3	0.335129	0.340623
4	0.191888	0.325698
5	0.623156	0.520855

F: > import > 1_6.py > ...

```python
import pandas as pd

lr_5 = pd.read_csv("LR_5.csv", encoding='gbk')
lr_5['利润率'] = (lr_5['B001000000'] / lr_5['B001100000'])
lr_5['资产负债率'] = (lr_5['A002000000'] / lr_5['A001000000'])

lr_new = lr_5[
    (lr_5['利润率'].between(-3, 3)) &
    (lr_5['资产负债率'].between(-3, 3))
]

lr_new.to_csv("LR_new.csv", encoding='gbk', index=False)

print("处理后的数据行数和列数：", lr_new.shape)
print("前5个企业的利润率和资产负债率")
print(lr_new[['利润率', '资产负债率']].head())

```

任务 1.6 部分代码展示

2 财务数据指标分析及可视化

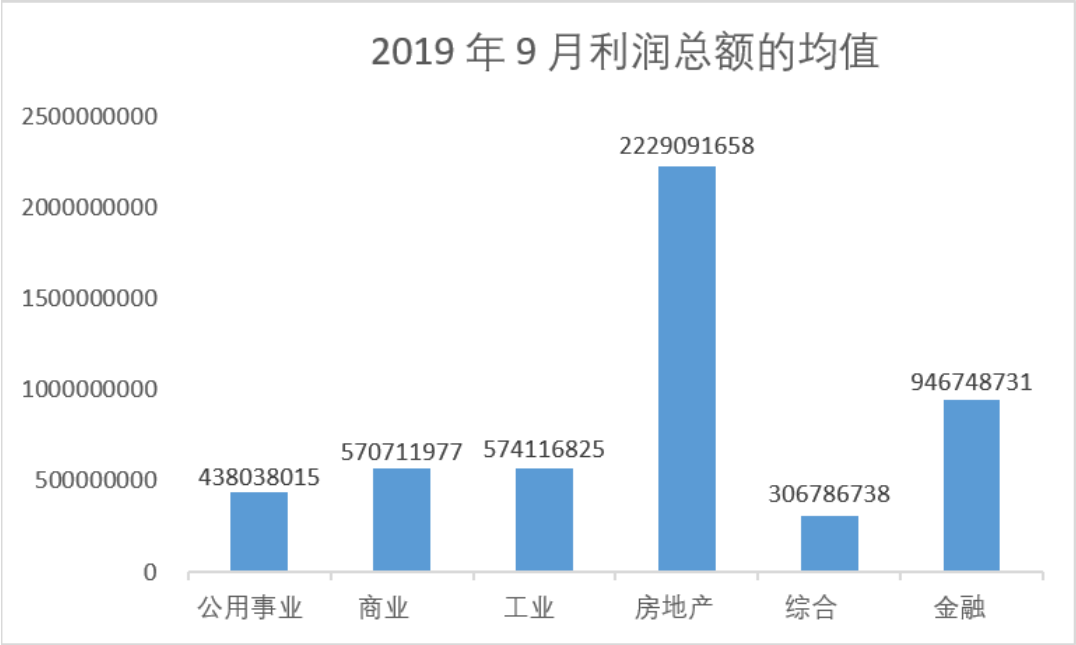
2.1 行业营业利润对比分析

2.1.1 各个行业大类 2019 年 9 月各行业大类的利润对比

小组根据“LR_new.csv”的统计结果，在 python 中执行相关操作。首先通过 excel 的筛选功能确定具体行业大类，随后导入 python 对各个行业大类 2019 年 9 月各行业大类的利润对比，获得具体数据后绘制柱状图。具体代码如下：

```
F: > import > 2_1_1.py > ...
1   import pandas as pd
2   df = pd.read_csv('LR_new.csv', encoding='utf-8')
3   df['Accper'] = pd.to_datetime(df['Accper'])
4
5   # 选择2019年9月的数据
6   september_data = df[(df['Accper'].dt.year == 2019)
7                       & (df['Accper'].dt.month == 9)]
8
9   # 计算输出B001000000列的均值
10  result = september_data.groupby('Indnme')['B001000000'].mean()
11  pd.set_option('display.float_format', lambda x: '%.5f' % x)
12  print(result)
13
```

任务 2.1.1 代码展示



2019 年 9 月利润总额均值的柱状图

分析：根据 2019 年 9 月利润总额均值的柱状图，，从高到低分别是房地产、金融、工业、商业、公用事业、综合。房地产产业在利润总额的均值上遥遥领先于其他产业，甚至超过了排 1 名第二金融业均值的 2 倍；在剩下的产业中，金融产业排名均值的第二，超出其他产业较多；中游的公用事业、商业和工业利润总额的均值相差不大；综合产业为所有产业中均值最低。

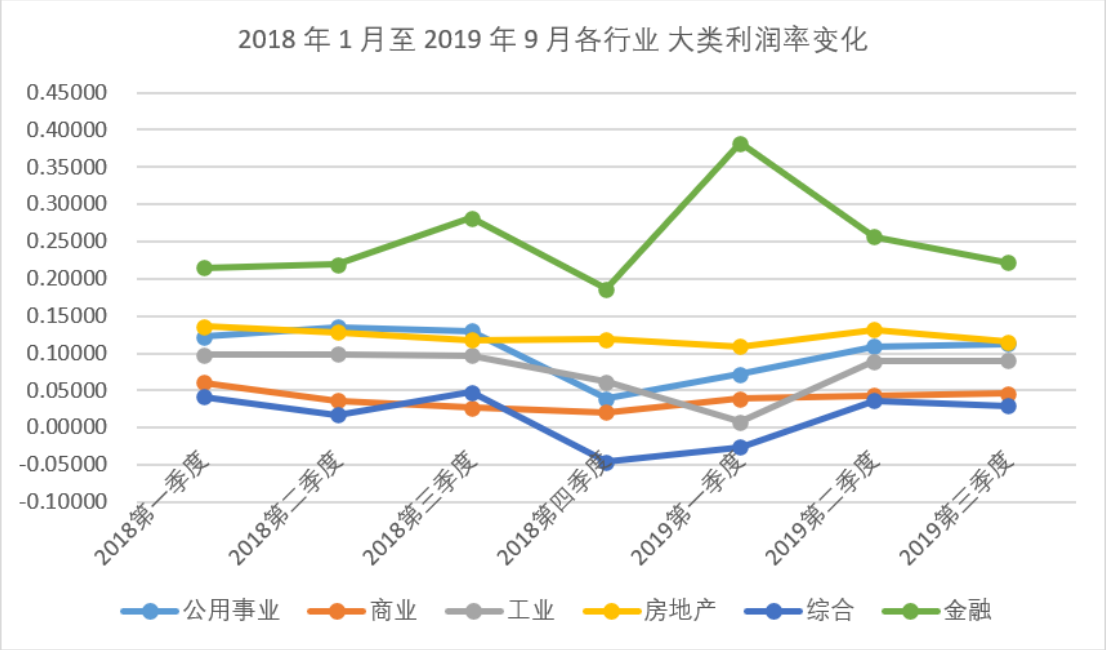
2.1.2 2018 年 1 月至 2019 年 9 月各行业大类利润率变化

同理，先对时间进行分类，2018 年为 4 个季度，2019 年为 3 个季度。随后统计各个季度利润率均值，即计算 B001000000 列的均值。下面给出 2019 年 9 月部分的核心代码供审阅。

```
F: > import > 2-1-2.py > ...
1   import pandas as pd
2
3   # 读取 Excel 文件
4   file_path = 'LR_new.csv'
5   df = pd.read_csv(file_path,encoding='utf-8')
6   df['Accper'] = pd.to_datetime(df['Accper'])
7
8   # 提取出 2019 年 9 月的数据
9   def Average_caculating(month):
10      month_data = df[(df['Accper'].dt.year == 2018)
11                      & (df['Accper'].dt.month == month)]
12
13      result = month_data.groupby('Indnme')['利润率'].mean()
14      pd.set_option('display.float_format', lambda x: '%.5f' % x)
15
16      return result
17
18  for i in range(1,13):
19      print(f'第{i}月的利润率平均值为{Average_caculating(i)}')
20
```

任务 2.1.2 部分代码展示

对其余年份和季度的代码在此省略。下面给出 2018 年 1 月至 2019 年 9 月各行业大类利润率变化的折线图。



分析：

- 金融行业利润率波动幅度较大，但均位于各行业大类的第一名且距离第二名差距很大。在 2019 年的第一季度到达最高的利润率接近 40%。
- 房地产行业利润率波动幅度很小，基本位于利润率排名的第二位，在 2018 年前 2 季度和公用事业基本持平，在 2018 年第四季度及 2019 年前 2 季度均领先于公用事业。
- 公用事业利润率在 2018 年前二季度保持平稳，但在 2018 年第三和第四季度之间遭遇大幅度下滑，但没用低于 4%的利润率。第四季度开始回升，随后到 2019 年第三季度逐步攀升回到了 2018 年第一季度的利润率水平。
- 工业利润率在 2018 年前二季度保持平稳，但在 2018 年第三季度开始处于下滑状态，到 2019 年第一季度达到最低约 1%的利润率，随后在 2019 年第一季度和第二季度之间快速回升，大体回到了 2018 年第一季度的利润率水平。
- 商业利润率平稳但偏低，利润率一直处于 5%左右，处于六大产业下游。

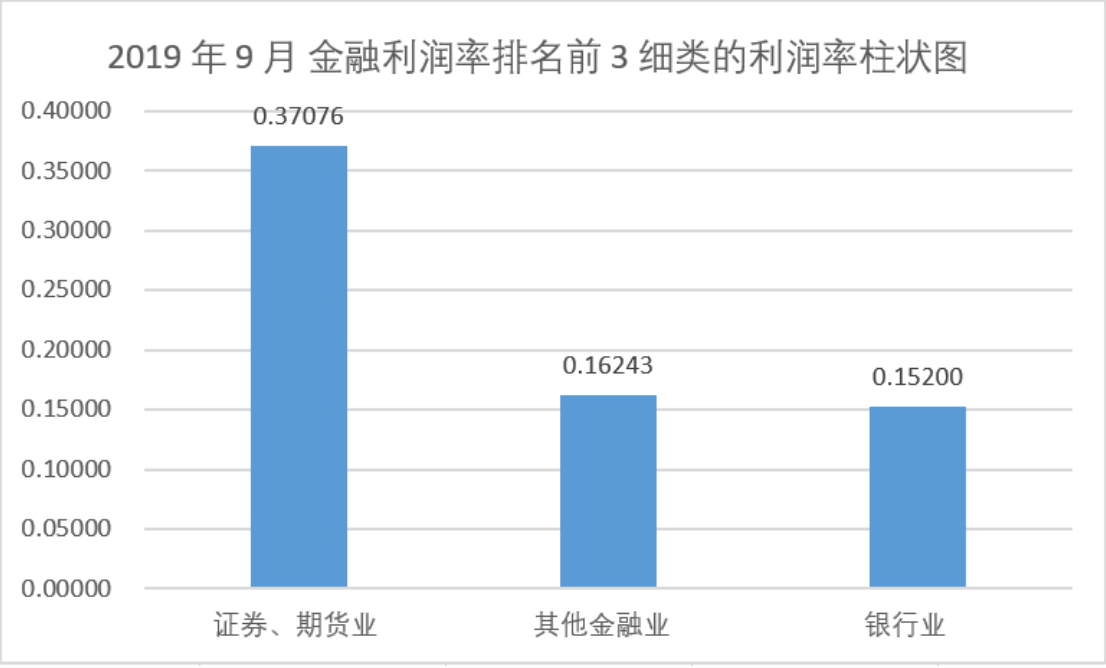
- 综合产业的利润率波动较大且一直处于各大产业的垫底水平。波动情况和公用事业基本一致，但在 2018 年第三季度到 2019 年第 1 季度的低谷期一直处于亏损状态。

2.2 行业与企业营收分析

2.2.1 2019 年该行业各细类利润率对比

读取“LR_new.csv”，由任务 2.1 的结果可以直观看出，2019 年 9 月营业利润率均值排名第一的行业大类是金融。随后根据题目中的图表要求，同样使用 python 作为工具进行运算，统计分别统计金融不同细类 2019 年 9 月的利润率，同时将数据导入 excel 中进行绘制。

经过程序统计后，绘制图表。



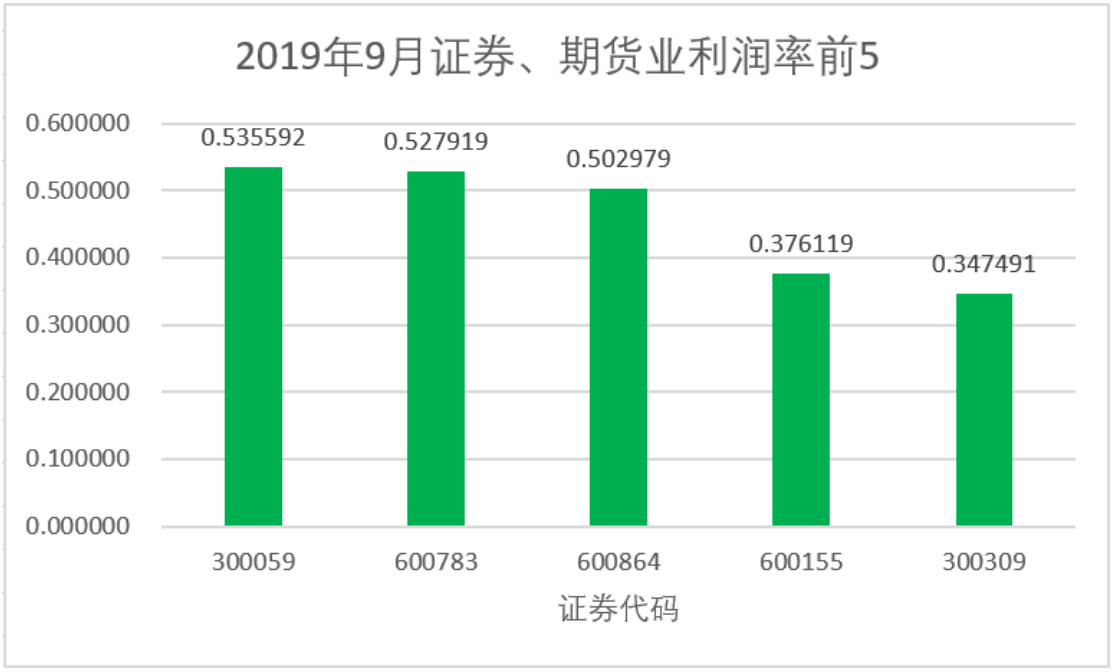
分析：从图中可以看出证券、期货业以绝对的优势排名第一，其他金融业和银行业分别是第二位和第三位。证券、期货业的利润率相较于其他两个细类有着巨大的优势，同时第二位和第三位的利润率差距较小。

2.2.2　2019 年该行业利润率排名第 1 细类的企业利润率对比

从 2.2.1 的结论可以看出，金融行业大类中利润率排名第一细类为证券、期货类。再次使用 python 的 pandas 库对证券、期货类排名前 5 的企业利润率进行读取，并导出到 excel 表格中进行绘制图表。下面给出部分代码。

```
df = pd.read_csv('LR_new.csv', encoding='utf-8')

df['Accper'] = pd.to_datetime(df['Accper'])

# 选择2019年9月的数据
september_data = df[(df['Accper'].dt.year == 2019)
                    & (df['Accper'].dt.month == 9)]
september_data = september_data[september_data['Nindnme'] == '证券、期货业']

result = september_data['利润率'].sort_values(ascending=False)

# 输出结果
print(result)
```

根据统计绘制表格：



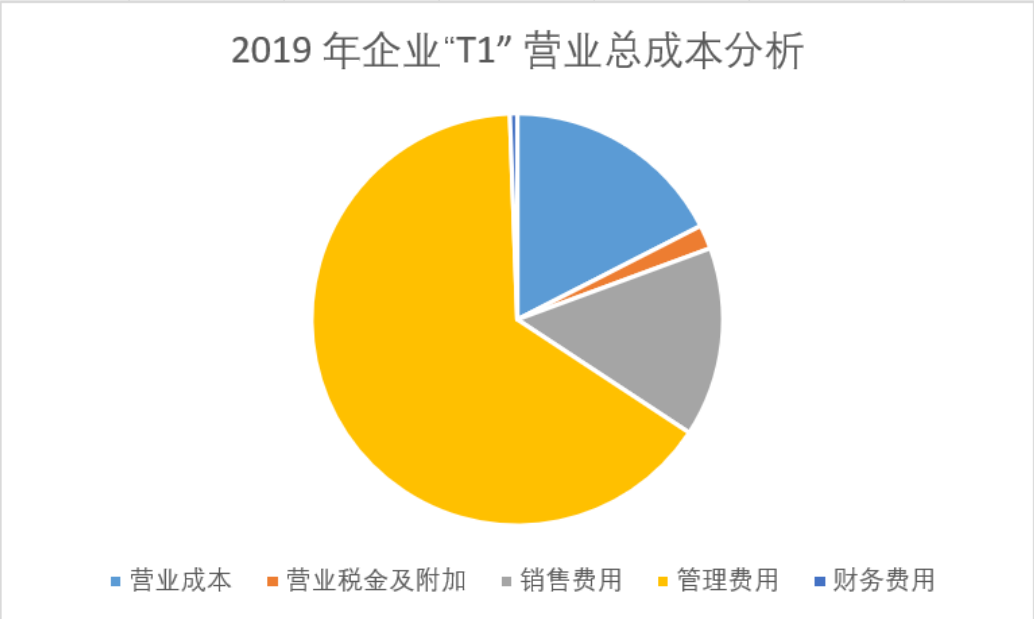
分析：2019 年 9 月证券、期货业利润率前 5 如图所示。可以看出证券、期货业的领头企业利润率非常高，第一名到第三名之间利润率差距很小，第三名和第四名有比较大的断层，第五名和第四名利润率的差距小。说明证券、期货业的利润率差距在总体上差别不大。

2.2.3 2019 年企业“T1”营业总成本分析和 2019 年企业“T1”经营情况分析

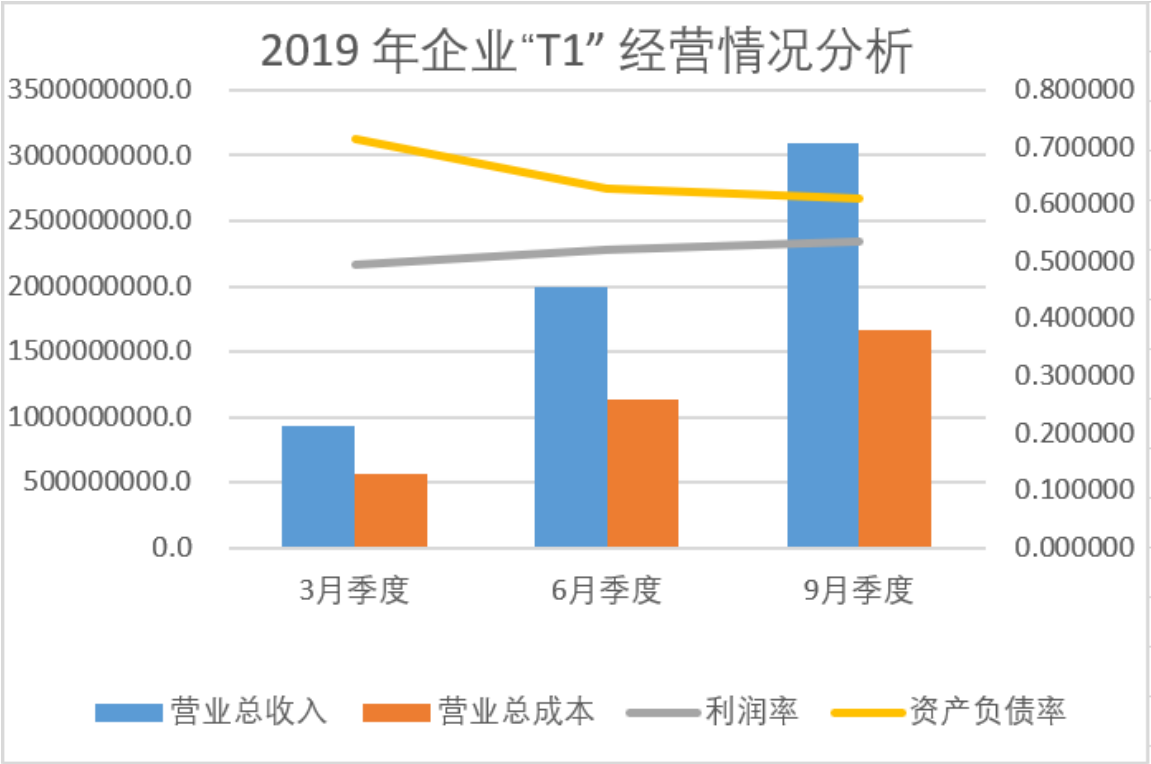
根据上表的统计结果，证券代码为 300059 的企业为“T1”企业。

绘制证券代码为 300059 的企业在 2019 年 9 月财务报表的营业成本、营业税金及附加、销售费用、管理费用、财务费用的饼图。

分析：根据饼图可以直观看出，管理费用超过了企业营业总成本的 60%，属于最大的支出部分，远远超越了营业成本和销售费用。营业成本和销售费用位居第二大成本和第三大成本，均占到了 15%左右。倒数第二的营业税金和最少成本占比的财务费用仅仅占据了 2.4%左右。



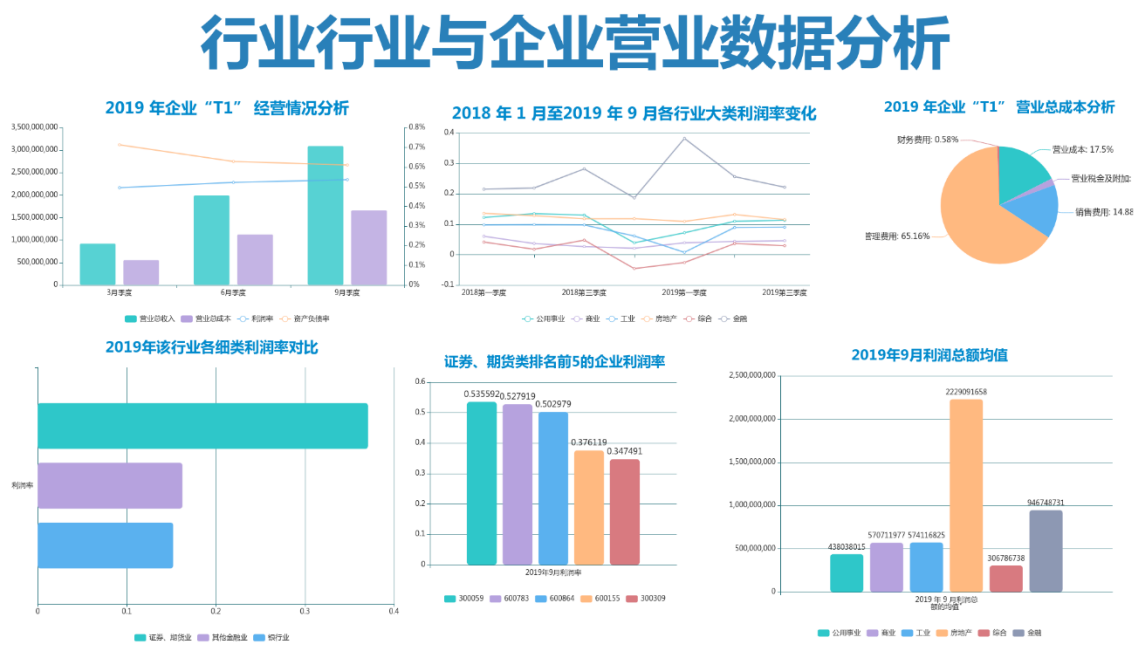
同时在一张图中绘制企业“T1”2019 年 3 月、6 月、9 月三个季度营业总收入、营业总成本的柱状图，绘制利润率、资产负债率变化的折线图。



分析：该企业在 2019 年 3 月季度的相对较高的负债率和较低的利润率，企业“T1”的营业总收入在 3 月季度到 9 月季度中以非常快的速度提升，利润率以平稳的速度攀升，同时资产负债率在 3 月季度到 6 月季度有明显的下降，到 9 月季度逐渐趋于平稳。总体来看，9 月季度的营业总收入已经来到了超过 3 月时的 3 倍。而营业总成本也随着公司的扩大而逐步增加，来到了 3 月季度时总成本的 3 倍左右。利润率的不断提高加上公司体量的快速扩大，说明该公司在 2019 年前半年的经营情况快速提升，随后逐渐稳步提升。

2.3 行业与企业营业数据分析——可视化大屏设计

小组将任务 2.1 和任务 2.2 所列的 6 张图制作成一个大屏，按照题意将大屏命名为“行业与企业营业数据分析”。



3 企业利润预测及财务造假识别

3.1 计算指标与利润总额的相关性，挑选指标

小组先使用了 Pearson 法与 Spearman 法分别计算了相关性。通过观察数据分布来决定更合理的指标。

其中计算 Pearson 方法的部分代码：

```
# 计算相关性
correlation = data.corr(method="pearson").abs()
# 选择与利润总额相关性最高的 5 个指标
top_5_correlated = correlation['LRZE'].nlargest(6)[0:]
print(top_5_correlated)
```

1.皮尔逊相关系数（Pearson correlation）

当两个变量都是正态连续变量，且两者之间呈线性关系时，则可以用Pearson来计算相关系数。**取值范围[-1,1]**。计算公式如下：

$$\rho(x,y)=\frac{\operatorname{cov}(x,y)}{\sigma(x)\cdot\sigma(y)}=\frac{E[(x-\mu_x)(y-\mu_y)]}{\sigma(x)\cdot\sigma(y)}$$

Pearson 方法计算结果：

YYSR 0.782726
YWFY 0.772832
YYCB 0.737736
YYSJJFJ 0.565440
ZCJZSS 0.238524

2、斯皮尔曼相关系数（Spearman correlation）

1、秩相关系数

秩相关系数（Coefficient of Rank Correlation），又称等级相关系数，反映的是两个随机变量的变化趋势方向和强度之间的关联，是将两个随机变量的样本值按数据的大小顺序**排列位次**，以各要素样本值的**位次代替实际数据**而求得的一种统计量。它是反映等级相关程度的统计分析指标，常用的等级相关分析方法有**Spearman相关系数**和Kendall秩相关系数等。主要用于数据分析。斯皮尔曼相关系数被定义成等级变量之间的皮尔逊相关系数。

2、使用条件

- 数据为非线性或非正态
- 至少有一组数据为**等级类型**，如排名，位次
- 数据中有**异常值**或错误值，斯皮尔曼相关系数对于异常值不太敏感，因为它基于排序位次进行计算，实际数值之间的差异大小对于计算结果没有直接影响

3、求相关系数

较为常用简单的计算公式如下所示：

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)}$$

- d_i 表示第i个数据对的位次值之差
- n 总的观测样本数

Spearman 方法计算结果:
YYSR 0.609192
YYSJJFJ 0.608179
YWFY 0.570395
YYCB 0.510354
ZCBCL 0.475194

可以看到利润总额与各项指标之间的关系应更趋向于线性相关，但是 Pearson 相关系数计算公式要求适用场景是呈正态分布的连续变量。显然题目中所给数据并不符合，而 spearman 相关系数计算公式并没有 Pearson 相关系数计算公式要求严格，而且计算结果更为均匀合理，所以小组采用 Spearman 相关系数计算公式来进行计算。

故相关性最高的 5 个指标为：YYSR、YYSJJFJ、YWFY、YYCB、ZCBCL

3.2 建立模型预测利润总额

根据 3.1 的分析可知，该利润总额满足简单线性关系：

LRZE= a*YYSR+b*YWFY +c*YYCB +d*YYSJJFJ +e*ZCBCL

故建立线性回归模型，使用线性回归模型的 COEF 属性来获取系数。

```
# 读取训练数据
df = pd.read_csv('financial_data.csv')

# 选择特征和目标变量
features = df[['YYSR', 'YWFY', 'YYCB', 'YYSJJFJ', 'ZCBCL']]
target = df['LRZE']

# 将数据分为训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, random_state=42)

# 数据标准化
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

通过分离训练集和测试集，并将数据标准化，完成了对数据集的预处理。

```
# 创建并训练线性回归模型
linear_model = LinearRegression()
linear_model.fit(X_train_scaled, y_train)

# 获取系数
coefficients = linear_model.coef_
pd.set_option('display.float_format', lambda x: '%.5f' % x)
print("Coefficients (a, b, c, d, e):", coefficients)

# 在测试集上进行预测
y_pred = linear_model.predict(X_test_scaled)
```

建立了线性回归模型，获取系数并在测试集上对总额进行预测

结果为 Coefficients (a, b, c, d, e): [1.79070619e+10 ； -1.60816790e+09;

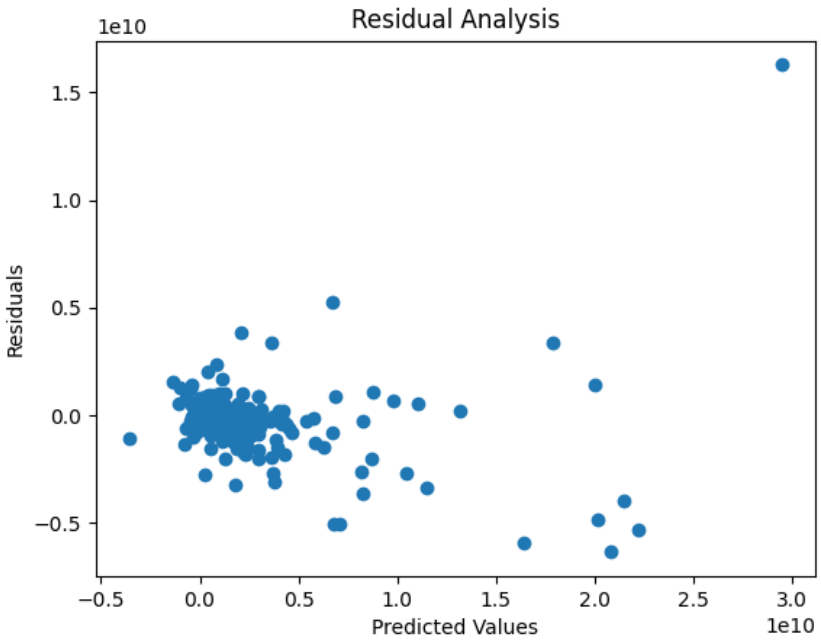
-1.51707250e+10 ； -2.12693412e+08 ； 3.50513266e+07]
R-squared on Test Set: 0.87367

```
# 评价模型性能
r2 = r2_score(y_test, y_pred)
print(f'R-squared on Test Set: {r2:.5f}')
pd.set_option('display.float_format', None)

# 残差分析
residuals = y_test - y_pred
plt.scatter(y_pred, residuals)
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Analysis')
plt.show()
```

评价模型的性能并进行残差分析

R-squared on Test Set: 0.87367 接近 1

R-squared (R^2): R-squared 是一个统计度量，用于衡量模型对因变量变异性的解释程度。**R-squared 的取值范围在 0 到 1 之间，越接近 1 表示模型对观测数据的解释越好。**在本问下，R-squared 值相对较高 (0.87)，表明模型能够解释目标变量的大部分变化。



残差分析：残差随着预测值的增加而有规律地变化，呈现锥形，模型拟合较好

```
# 数据标准化（使用之前训练集的标准化器）
test_features_scaled = scaler.transform(test_features)

# 使用模型进行预测
predicted_LRZE = (
    coefficients[0] * test_features_scaled[:, 0] +
    coefficients[1] * test_features_scaled[:, 1] +
    coefficients[2] * test_features_scaled[:, 2] +
    coefficients[3] * test_features_scaled[:, 3] +
    coefficients[4] * test_features_scaled[:, 4]
)
```

对数据进行标准化并调用模型开始预测，核心部分代码已作附在论文末尾。

TICKER_SYMBOL	LRZE
4953174	73592971.02203822
4961537	47816892.75837927

4962538	-260682214.9154947
4968740	- 231340004.26481417
4973917	-320213539.9146815
4978589	-39916094.073338665
4978721	-294688770.1332381
4986535	-285294458.0184807
4990739	-125963875.13080499
4990942	-293451929.07646364

*模型改进和推广：

1、模型可能引发的过度拟合问题

为了尽可能减轻模型引发的过度拟合问题和其他由于过度拟合导致的数据偏差，可以采用正则化的处理方式。正则化是一种用于控制模型复杂性的技术，常用于线性回归等模型，以防止过拟合。两种常见的正则化方法是岭回归 (Ridge Regression)和 LASSO 回归 (Least Absolute Shrinkage and Selection Operator Regression)

岭回归（Ridge Regression）：

岭回归通过在损失函数中添加 L2 范数的平方（模型权重的平方和）来实现正则化。岭回归的损失函数为：

$$L(\theta) = \text{MSE} + \alpha \sum_{i=1}^{n} \theta_i^2$$

其中，α是正则化强度的超参数，增大α将使得模型的权重更趋向于零，减小了模型的复杂性。

LASSO 回归：

LASSO 回归通过在损失函数中添加 L1 范数的绝对值和（模型权重的绝对值和）来实现正则化。LASSO 回归的损失函数为：

$$L(\theta) = \text{MSE} + \alpha \sum_{i=1}^{n} |\theta_i|$$

同样，α是正则化强度的超参数。与岭回归不同，LASSO 回归的正则化项倾向于使一些权重变为零，因此可以用于特征选择。

2、仅使用 R-squared 可能导致的评价性能过于单一问题

尽管 R-squared 是一个常用的评价指标，但在实际应用中，仅依赖一个指标可能不足以全面评估模型。你可能还需要考虑其他指标，例如均方根误差 (Root Mean Squared Error，RMSE) 等，以更全面地了解模型性能。

3.3 筛查企业财务数据识别涉嫌财务造假企业

一、算法选择

使用流动比率（LDBL）、资产负债率（ZCFZL）、存货周转率（CHZZL）、资产报酬率（ZCBCL）和应收账款周转率（YSZKZZL）这些特征来预测财务数据是否涉嫌财务造假（FLAG）。这是一个二分类问题，属于监督学习问题，我们采用随机森林算法来求解。

随机森林（Random Forest）是一种集成学习（Ensemble Learning）方法，它通过构建多个决策树来提高模型的性能和泛化能力。随机森林属于Bagging（Bootstrap Aggregating）类型的集成学习方法，它通过对训练数据

的随机抽样和对特征的随机选择来创建多个决策树，并最终通过投票或平均来得出综合的预测结果。

```
# 读取财务数据
df = pd.read_csv('financial_data.csv')

# 选择特征和标签
features = df[['LDBL', 'ZCFZL', 'CHZZL', 'ZCBCL', 'YSZKZZL']]
labels = df['FLAG']

# 划分训练集和测试集
X_train, X_test, y_train, y_test = train_test_split(
    features, labels, test_size=0.2, random_state=42)

# 构建随机森林分类器
model = RandomForestClassifier(n_estimators=100, random_state=42)

# 训练模型
model.fit(X_train, y_train)

# 在测试集上进行预测
predictions = model.predict(X_test)
```

二、模型评价

```
# 评估模型性能
accuracy = accuracy_score(y_test, predictions)
conf_matrix = confusion_matrix(y_test, predictions)

print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)
```

其中:

Accuracy:

1.0

Confusion Matrix:

$$\begin{bmatrix} 2187 & 0 \\ 0 & 12 \end{bmatrix}$$

此种情况在训练集上的表现非常好，准确率达到了 1.0，也就是 100%。这说明在训练集上模型完美地拟合了数据。

此外，对于某些问题，数据中可能存在类别不平衡的情况，这也可能导致模型在训练集上表现很好，但在测试集上表现不佳。本题的混淆矩阵显示了训

练集上的完美预测，但小组认为也可以进一步考虑查看模型在测试集上的性能，特别是在负类别（FLAG=0，即非财务造假）上的性能。

三、结果

证券代码：4897311 涉嫌财务造假 （FLAG:1）

四、模型的改进与推广：

1、调整模型超参数： 通过调整随机森林的超参数，例如树的数量（estimators）、树的最大深度（max_depth）、每棵树的最小样本拆分数（min_samples_split）等，来优化模型性能。可以使用交叉验证来选择最佳的超参数组合。

2、解决不平衡类别： 如果数据中存在类别不平衡，即某一类别的样本数量远远多于另一类别，可以考虑采用一些方法来处理不平衡，如过采样、欠采样或使用加权类别。