标题：基于 Pandas+Seaborn+Matplotlib 居民健康状况数据可视化和基于 Sklearn 支持向量机分类模型的居民疾病状况预测

摘要：

随着现代医疗水平和现代计算机科学、人工智能、数据分析的不断发展，医院里也积累了大量的病人病历信息数据。通过大量信息的整合和修改，对这些数据进行整理与分析，有助于优化医疗资源配置、提升服务质量以及预测患者治疗效果。本数据集汇总了多位患者的基础信息、病情描述、就诊过程及医疗费用等关键内容，涵盖字段包括姓名、年龄、性别、血型、入院与出院日期、主治医生、所在医院、保险公司、账单金额、住院时长等[1]。本研究旨在使用 Python 支持库 Pandas+ Seaborn+ Matplotlib，通过数据分析，采集该数据集探索患者特征与医疗行为之间的潜在关系，分析例如年龄与账单金额的相关性、不同医疗条件下住院时长的差异等。通过数据分析及机器学习寻找不同字段之间的相关性，为智能医疗决策提供数据支持。结果表明：结果表明，通过对患者基础信息与医疗过程数据的深入分析，可以发现多种潜在的规律与关联关系。

Abstract：

With the advancement of modern medical technology, computer science, artificial intelligence, and data analytics, hospitals have accumulated a vast amount of patient medical record data. By integrating and refining this information, data analysis can help optimize the allocation of medical resources, improve service quality, and predict patient treatment outcomes. This dataset compiles key information from multiple patients, including basic demographics, disease descriptions, treatment processes, and medical expenses. It covers fields such as name, age, gender, blood type, admission and discharge dates, attending physician, hospital, insurance provider, bill amount, and length of stay. This study aims to explore the potential relationships between patient characteristics and medical behavior using Python libraries such as Pandas, Seaborn, and Matplotlib. Analyses include, for example, the correlation between age and billing amount, as well as differences in hospitalization duration under varying medical conditions. By applying data analysis and machine learning techniques, the study seeks to identify correlations among different variables and provide data-driven support for intelligent medical decision-making. The results show that in-depth analysis of patient demographics and medical process data reveals multiple underlying patterns and associations.

关键字:

Pandas、Seaborn、Matplotlib、Sklearn、数据分析、大数据技术、大数据、居民健康

0 引言

在计算机编程中，pandas 是用于数据操纵和分析的 Python 软件库。它建造在 NumPy 基础上，并为操纵数值表格和时间序列，提供了数据结构和运算操作。[2]在 Python 中，pandas 库的功能十分强大，它提供高性能的矩阵运算；可用于数据挖掘和数据分析，同时也提供数据清洗功能；支持类似 SQL 数据库的增、删、查、改等操作，并且带有丰富的数据处理函数；支持时间序列数据分析功能；支持灵活处理确实数据等。[3]Seaborn 是构建在 matplotlib 之上的数据可视化库，与 Python 中的 pandas 数据结构紧密集成。可视化是 Seaborn 的核心部分，可以帮助探索和理解数据。[4] Matplotlib 的功能则与 Seaborn 接近，也是帮助将数据进行可视化操作的第三方库函数。

随着科技的发展，我们每个人都是信息化时代的受益者。从医疗卫生的角度，医院积累了大量患者的诊断和就诊数据。我们对这些就诊数据进行数据分析和数据可视化，分析各个字段之间所存在的潜在联系。

综上所述，为了分析患者的客观信息及其医疗行为之间的关系，我们需要对此数据集进行数据分析，这里采用 python 中常见的第三方库，分别是 Pandas, Seaborn, Matplotlib。

1 Pandas, Seaborn, Matplotlib,scikit-learn 中各函数的功能、数学基础和基本原理

- 相关系数矩阵的绘制原理

首先我们明确相关系数的计算公式:

$$r(X,Y)=\frac{Cov(X,Y)}{\sqrt{Var[X]\,Var[Y]}}$$ [5]

从公式中我们知道，对于两个变量 x 和 y，我们首先需要知道二者之间的协方差系数 Cov(x,y)，然后计算二者的方差 Var(x),Var(y)，对于协方差系数，计算公式如下:

$$\operatorname{cov}(X,Y)=\mathrm{E}[(X-\mu)(Y-\nu)]$$ [6]

按照上述过程，我们可以计算出每两个字段之间的相关系数，将所有的相关系数绘制

在同一个矩阵中，我们就可以得到相关系数矩阵。在 python 中，我们可以使用以下代码求得相关系数矩阵：

```
correlation_matrix =df.corr()
```

但是不难发现其中会存在一些问题，比如，很多数据不是数值型的数据，而非数值类型的数据是很难求得相关系数的。因此在送入 corr()方法之前，我们应该先对 df 矩阵中所含的内容进行数据清洗，比如仅留下数值类型的数据进行分析，代码如下：

```
numeric_df = df.select_dtypes(include='number')
```

这样我们就可以通过 plt 绘制其相关系数矩阵。

- 箱线图与分位数

在将箱线图之前我们要先了解分位数的概念。

分位数（英语：Quantile），亦称分位点，是指用分割点（cut point）将一个随机变量的概率分布范围分为几个具有相同概率的连续区间。分割点的数量比划分出的区间少 1，例如 3 个分割点能分出 4 个区间。[7]

箱线图是通过 4 个分割点，找到其相应的四分位数和中位数，绘制出的一张能够反映数据重心分布和中位数的图片。

对于箱线图的创建，我们可以通过如下代码实现：

```
sns.boxplot(x='xx', y='yy', data=df, palette='Set2')

plt.xlabel('xx')

plt.ylabel('yy')
```

- 独立性检验的数学原理

在统计学中，独立性检验最常用的检验方法是卡方检验。该方法用于判断两个分类变量之间是否存在统计学关联。卡方检验的具体做法是是将观察到的数据与在变量相互独立的假设下所期望的数据进行比较，通过计算卡方 Chi2 统计量并与临界值比较，从而判断是否拒绝原假设，即判断变量间是否独立。

卡方检验常用以下公式计算：

	y1	y2	总计
x1	a	b	a+b
x2	c	d	c+d
总计	a+c	b+d	a+b+c+d

K2 = n (ad - bc) 2 / [(a+b)(c+d)(a+c)(b+d)]

其中 n=a+b+c+d 为样本容量[8]

因此我们在需要判断两个统计量是否具有关联时，可以通过这种方式进行判断，其实现代码如下：

```
def chi_square_test(a, b, c, d):
    n = a + b + c + d
    numerator = n * (a * d - b * c) ** 2
    denominator = (a + b) * (c + d) * (a + c) * (b + d)
    if denominator == 0:
        raise ValueError("分母为 0，无法进行卡方检验")
    chi2 = numerator / denominator
    return chi2
```

- 数据的分类

在数据建模过程中，我们可以使用机器学期对数据集进行分类。

第一步，我们要对数据集进行划分，划分是其中重要的一环。通常将原始数据划分为训练集和测试集。数据划分的原则应遵循样本独立性和分布一致性，避免数据泄露和过拟合。

第二步我们可以引入 SVM 等算法对数据进行分类。SVM，支持向量机，是在分类与回归分析中分析数据的监督式学习模型与相关的学习算法。给定一组训练实例，每个训练实例被标记为属于两个类别中的一个或另一个，SVM 训练算法建立一个将新的实例分配给两个类别之一的模型，使其成为非概率二元线性分类器。SVM 模型是将实例表示为空间中的点，这样映射就使得单独类别的实例被尽可能宽的明显的间隔分开。然后，将新的实例映射到同一空间，并基于它们落在间隔的哪一侧来预测所属类别。[9]

2 基于 Pandas, Seaborn, Matplotlib 的数据分析

我们这次将从多个方面对该数据文件进行分析

首先我们会通过 info 观察 csv 文件的形状，存有的字段和数据类型。

然后，我们可以通过 Matplotlib 对文件进行预处理，一方面可以去尝试判断该文件中

是否有字段出现缺失，另一方面，我们需要对可能存在的重复值进行去重操作。

随后，我们绘制相关系数热力图，试图从中寻找可能存在关系的字段，但是由于本数据集中的数值型数据较少，所以通过 sns.heatmap 函数无法有效绘制非数值型字段之间的热力图。

之后我们可以对在之前相关系数热力图分析中，对出现的可能存在关系的字段进行数据分析。数据分析可以从如下几个维度进行：绘制散点图、箱线图、柱状图等；使用 Chi2 进行独立性检验；建立模型以通过已有的字段预测其他字段。

最后我们利用 scikit-learn 中的支持向量机 SVM，构建基于医疗数据集的支持向量机分类模型，并对该模型进行评价。

3 利用 python 对居民的健康数据进行数据预处理

收集到的数据存在如下字段。

英文字段（English）	中文字段（简体中文）
Name	姓名
Age	年龄
Gender	性别
Blood Type	血型
Medical Condition	病情情况
Date of Admission	入院日期
Doctor	主治医生
Hospital	医院
Insurance Provider	保险公司/保险提供方
Billing Amount	账单金额
Room Number	病房号
Admission Type	入院类型（如急诊/预约等）
Discharge Date	出院日期
Medication	药物/用药情况
Test Results	检查结果
Length of Stay	住院时长

- 数据导入

首先导入数据集并使用 info 查看具体数据，程序运行结果如图所示：

```
(base) fang50253@MacBook_Pro Healthcare_Dataset % python3 -u "/Users/fang50253/Desktop/Fi
les/Documents/NJFU_My_Github/NJFU_CS_Note/24-25-2大数据技术与应用/写作业/Healthcare_Datas
et/code_final.py"
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55500 entries, 0 to 55499
Data columns (total 16 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                55500 non-null  object
 1   Age                 55500 non-null  int64
 2   Gender              55500 non-null  object
 3   Blood Type          55500 non-null  object
 4   Medical Condition   55500 non-null  object
 5   Date of Admission   55500 non-null  object
 6   Doctor              55500 non-null  object
 7   Hospital            55500 non-null  object
 8   Insurance Provider  55500 non-null  object
 9   Billing Amount      55500 non-null  float64
 10  Room Number         55500 non-null  int64
 11  Admission Type      55500 non-null  object
 12  Discharge Date      55500 non-null  object
 13  Medication          55500 non-null  object
 14  Test Results        55500 non-null  object
 15  Length of Stay      55500 non-null  int64
dtypes: float64(1), int64(3), object(12)
memory usage: 6.8+ MB
None
```

首先大致观察数据和列标签的形态，接着运用相关函数查看数据的基本信息。由此可知，该数据集共包含 55,500 行、16 列，记录了医院接诊的大量患者的患病和治疗信息其中，“Billing Amount”一列为浮点型，表示患者的账单金额；“Age”、“Room Number”以及“Length of Stay”为整型，分别记录了患者的年龄、住院病房号和住院的时长；其余 12 列均为对象（object）类型，对应字符串数据。例如，“Date of Admission”和“Discharge Date”表示患者的入院和出院日期；“Gender”、“Blood Type”、“MedicalCondition”等列提供了患者的基本信息；“Doctor”和“Hospital”表示接诊医生及其所在医院；而“Insurance Provider”、“Admission Type”、“Medication”和“Test Results”等则涵盖了患者的医保类型、入院方式、用药情况及检测结果等。例如，数据集中采用保险公司的具体名称，比如 UnitedHealthcare、Blue Cross、Aetna、Cigna、Medicare 来表示保险的提供商。

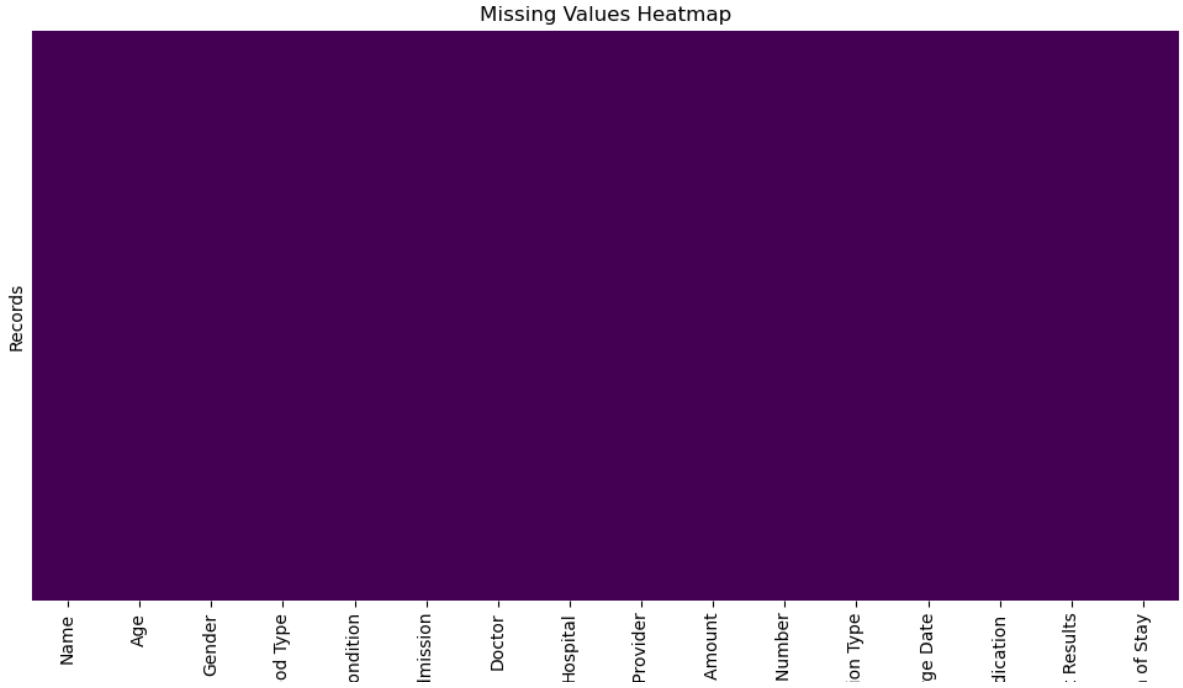
最后输出的该 csv 文件共占用存储空间 6.8MB。

- 查看数据缺失情况

下面我们采用 matplotlib.pyplot 库中的绘图函数，绘制一张缺失值热图。显然，在这段数据中没有缺失值，如果存在缺失值我们还需要对缺失值进行补全操作。代码和绘制的图片如下：

```
# 2.使用 Matplotlib 查看是否存在数据缺失

plt.figure(figsize=(12, 6))

sns.heatmap(data.isnull(), cbar=False, cmap='viridis', yticklabels=False)
```

```
plt.title('Missing Values Heatmap')

plt.xlabel('Columns')

plt.ylabel('Records')

plt.savefig('missing_values_heatmap.png')
```



从最后输出的热力图上来看，所有字段的数据均未发生数据缺失，因此本数据的每一条记录都是相对完整的，无需对缺失值进行处理，或者插值。

- 重复值分析

重复值分析：经过分析后我们发现原数据不存在重复值，代码如下：

```
# 3.重复值分析

duplicate_rows = data.duplicated()

num_duplicates = duplicate_rows.sum()

print(f"Number of duplicate rows: {num_duplicates}")

if num_duplicates > 0:

    print("Duplicate rows:")

    print(data[duplicate_rows].head())

data_cleaned = data.drop_duplicates()
```

print(data_cleaned.info())[10]

清除重复数据前，数据集的尺寸为 55500 行×16 列；清除后，数据集的大小仍为 55500 行×16 列，这表明本数据集没有重复的数据值，无需对重复值进行去重操作。

4. 利用 python 对居民的健康数据进行数据分析

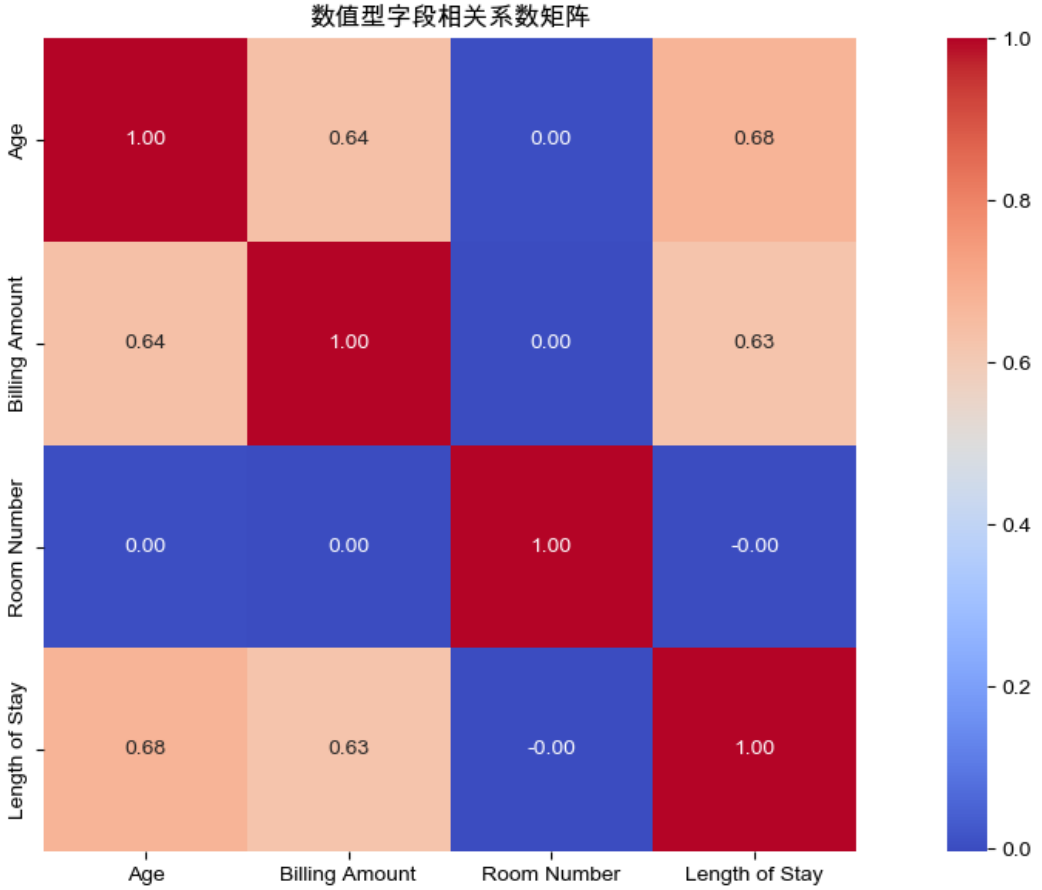
- 绘制相关系数热力图

为了解各个字段之间的相关性，我们可以绘制相关系数热力图矩阵，通过观察各个字段之间的相关系数对数据做出初步分析。由于 object 类型的数据无法直接计算相关系数，因此我们使用 numeric_df = data_cleaned.select_dtypes(include='number')提取数值型的字段，对数值型的数字进行数据分析，代码如下：

```
# 4.数据相关系数热力图

plt.figure(figsize=(12, 6))

numeric_df = data_cleaned.select_dtypes(include='number')

correlation_matrix = numeric_df.corr()

sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f",square=True)

plt.title("数值型字段相关系数矩阵")

plt.tight_layout()

plt.savefig('correlation_matrix_heatmap.png')
```

从图中我们看到 Room，即房间号与年龄、住院时长和账单天数之间无相关性，符合常规的认知。年龄和账单金额、账单金额和住院天数、住院天数和年龄之间都有中等偏强的相关性，因此我们在下面的数据分析中可以着重分析这些具有较强相关性的字段之间的规律。

但是由于原始数据中的数值型数据数量较为有限，为了更加准确地展现各个字段之间的相关性，我们还需要想办法对非数值型的数据进行比较，以此来确定他们之间存在的联系。



- 统计账单金额箱线图

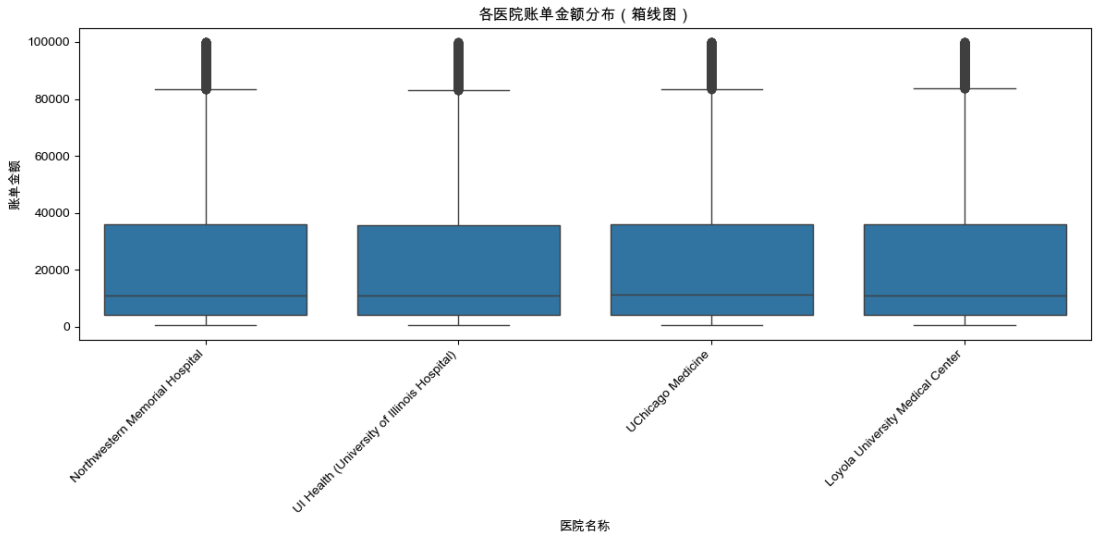
通过观察 csv 文件发现，这个数据集中的数据主要来源于 4 家医院，分别对这 4 家医院的账单金额进行统计，并绘制箱线图，代码如下：

```
# 5.将数据按照医院划分，统计账单金额的箱线图

plt.figure(figsize=(12, 6))

sns.boxplot(data=data_cleaned, x='Hospital', y='Billing Amount')

plt.title('各医院账单金额分布（箱线图）')

plt.xlabel('医院名称')

plt.ylabel('账单金额')

plt.xticks(rotation=45, ha='right')

plt.tight_layout()

plt.savefig('billing_amount_boxplot_by_hospital.png')

quantiles = data_cleaned.groupby('Hospital')['Billing Amount'].quantile([0.25, 0.5, 0.75])
```

```
quantiles = quantiles.unstack()

quantiles.columns = ['25%', '50%', '75%']

quantiles = quantiles.round(2)

print("各医院账单金额的分位数统计：")

print(quantiles)
```

我们得到了如下结果：

```
各医院账单金额的分位数统计：
                                          25%       50%       75%
Hospital
Loyola University Medical Center          4156.39  10907.32  35991.60
Northwestern Memorial Hospital            4196.10  11086.84  35897.51
UChicago Medicine                         4223.68  11201.67  35875.36
UI Health (University of Illinois Hospital) 4200.32  11113.24  35743.71
```



由箱线图可知，接受统计的各个 4 家医院在账单金额上接近，且账单金额的 25、50、75 百分位数也比较接近。没有表现出显著差异，而仅仅通过数值型数据的相关系数矩阵是很难体现出来的。

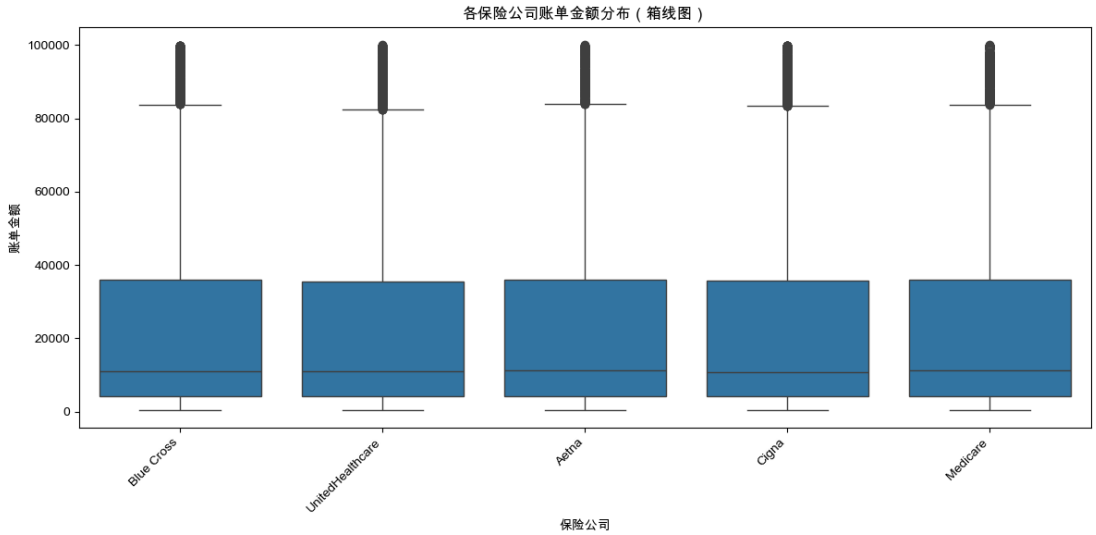
- 对不同保险公司的账单金额作箱线图

同样，我们可以对保险公司、检查结果和用药情况做出类似分析：

对于不同的保险公司提供商，其账单金额也没有表现出显著差异。

各保险公司账单金额的分位数统计：

Insurance Provider	25%	50%	75%
Aetna	4229.19	11268.13	36077.42
Blue Cross	4203.72	11023.21	36026.90
Cigna	4155.51	10871.88	35850.08
Medicare	4191.06	11162.99	36002.41
UnitedHealthcare	4182.01	11019.86	35487.10



- 不同的用药情况和诊断难度对账单金额的影响

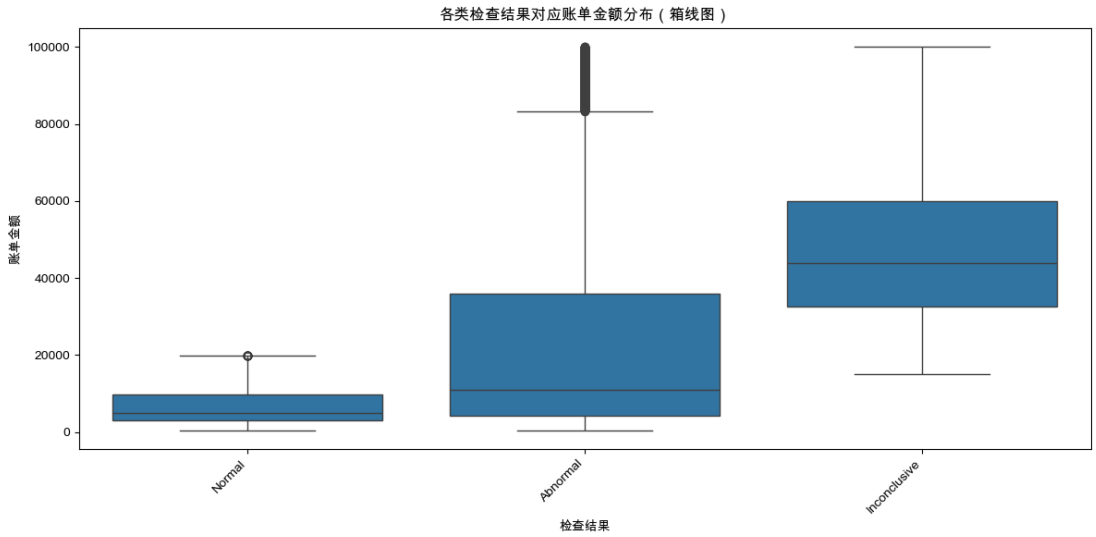
但是面对不同的检查结果，一般的检查结果和不正常的检查结果有着显著的差距，面对无法确诊的病症，其账单即费用远高于一般的病症，这也意味着患者或者保险公司将会花费更多的钱。

不同的用药情况同样会带来账单金额的巨大差异，部分药品的价格高昂，部分药品的价格低廉，从箱线图中我们也看到了不同药品之间的显著差异。部分患者在使用价格高昂的药物时，其保险所支付的金额远大于其他的疾病，甚至能达到部分疾病的十多倍之多。

这项数据对保险公司来说有重要意义，更高的金额代表保险公司将会支出更多的费用，对于这些价格高昂的药品，保险公司可以制定特殊的政策和条款来限制其赔款金额，或者设立单独的保险用于保障这些治疗费用高昂的疾病。

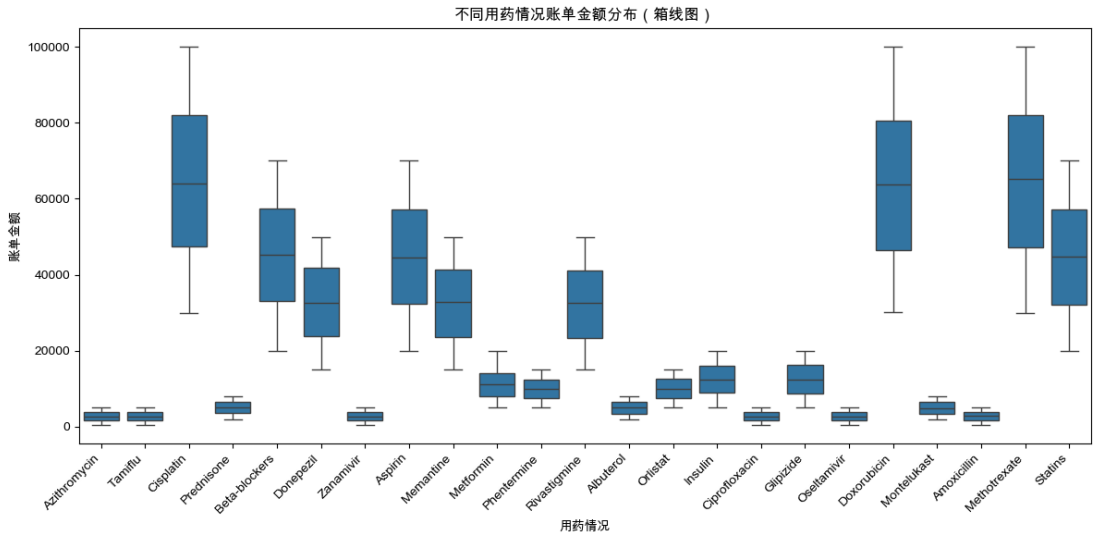
各类检查结果账单金额的分位数统计：

Test Results	25%	50%	75%
Abnormal	4213.85	11083.88	35868.45
Inconclusive	32479.67	43945.31	59894.91
Normal	2955.31	5017.35	9716.74



不同用药情况账单金额的分位数统计：

Medication	25%	50%	75%
Albuterol	3492.35	5064.39	6563.02
Amoxicillin	1643.38	2814.93	3905.17
Aspirin	32482.24	44493.24	57227.35
Azithromycin	1591.65	2683.62	3896.23
Beta-blockers	33203.89	45161.95	57396.07
Ciprofloxacin	1606.23	2681.63	3838.34
Cisplatin	47560.99	63908.49	82047.13
Donepezil	23782.24	32543.88	41889.44
Doxorubicin	46399.67	63652.10	80515.99
Glipizide	8823.33	12408.59	16196.65
Insulin	8913.53	12493.41	16067.62
Memantine	23572.20	32869.48	41439.01
Metformin	8053.18	11090.41	13993.69
Methotrexate	47255.52	65313.36	81969.50
Montelukast	3412.07	4922.52	6450.77
Orlistat	7476.27	9986.42	12536.47
Oseltamivir	1597.19	2752.43	3857.72
Phentermine	7481.63	10042.49	12511.12
Prednisone	3644.36	5115.70	6637.96
Rivastigmine	23322.70	32687.75	41169.27
Statins	32006.38	44731.15	57231.33
Tamiflu	1615.64	2698.55	3908.39
Zanamivir	1621.63	2739.10	3861.51



- 性别和患病病情之间的关系

下面我们来分析另 2 个可能存在关系的，即性别和病情之间的关系。这次我们可以选择对两者进行独立性检验，即利用卡方检验计算二者的相关性，并计算 p 值。但是最终我们发现 p 值高达 0.8113，二者之间不存在显著关联。下面给出进行独立性检验的代码：

```
# 7.独立性检验性别 VS 病情

contingency_table = pd.crosstab(data['Gender'], data['Medical Condition'])

print("\n 交叉表（性别 vs. 病情）:")

print(contingency_table)

chi2, p, dof, expected = chi2_contingency(contingency_table)

print("\n 卡方检验结果:")

print(f"Chi2 统计量: {chi2:.4f}")

print(f"自由度: {dof}")

print(f"p 值: {p:.4f}")

alpha = 0.05

if p < alpha:

    print("结论：性别与患病种类之间存在显著统计关联。")

else:

    print("结论：性别与患病种类之间不存在显著统计关联。")
```

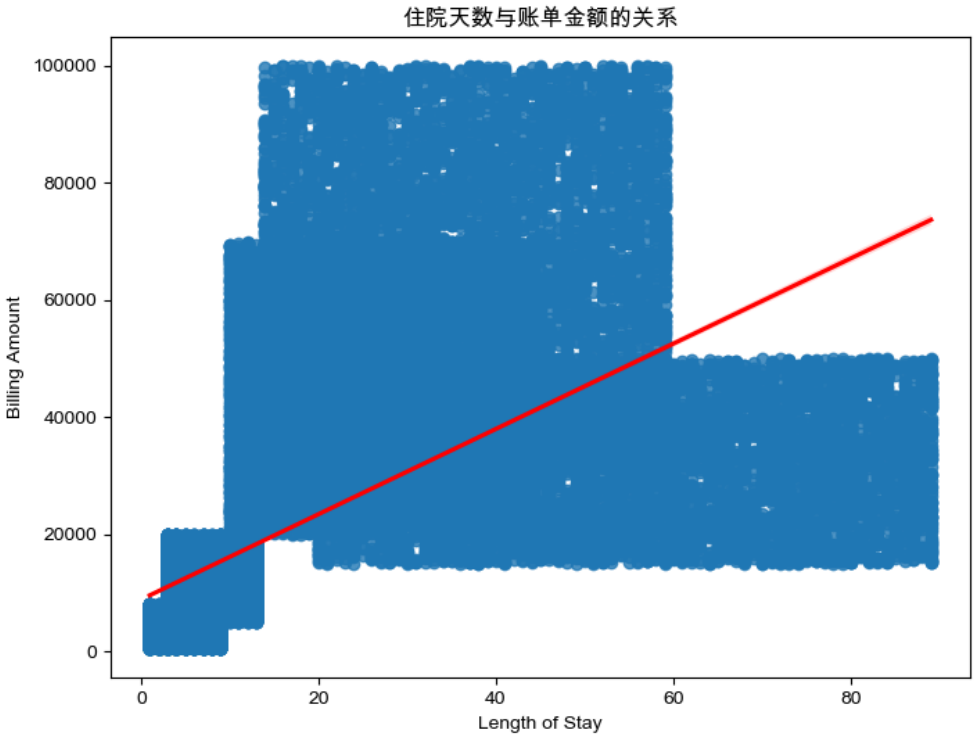
```
plt.figure(figsize=(10, 6))

sns.heatmap(contingency_table, annot=True, cmap="YlGnBu", fmt="d")

plt.title("性别 vs. 患病种类 - 交叉分布热力图")

plt.ylabel("性别")

plt.xlabel("患病种类")

plt.tight_layout()

plt.savefig("患病情况与性别的关系.png")
```

```
交叉表（性别 vs. 病情）：
Medical Condition  Alzheimer鈥檚  Asthma  Cancer  Diabetes   Flu  Heart Disease  Infections  Obesity
Gender
Female                    3479    3422    3506      3505  3531           3505        3459     3548
Male                      3382    3486    3434      3500  3515           3395        3387     3446

卡方检验结果：
Chi2统计量：3.7210
自由度：7
p值：0.8113
结论：性别与患病种类之间不存在显著统计关联。
```

独立性检验的 p 值相当大，说明两者之间不存在关联。

- 账单金额和住院天数之间的关系

接下来我们分析账单金额与住院天数的关系，不同于上面，这次我们采用散点图的方式来绘制，并且通过计算其回归方程。



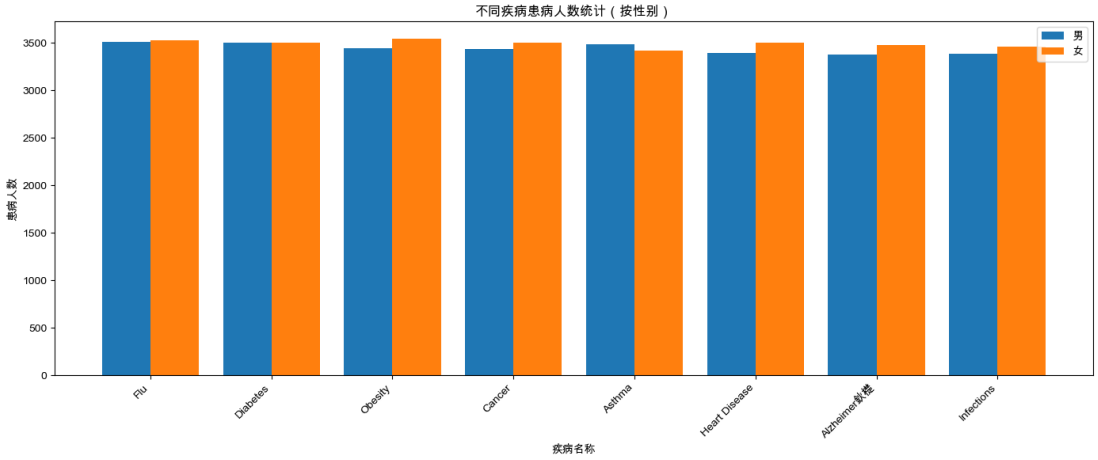
从散点图的角度上来看其线性相关的关系并不显著，治疗总费用和住院天数之间的表格之间呈现了多个方形拼凑的形状。虽然说程序给出了其线性拟合关系，但是从图上看我主观认为其相关性不显著，该拟合不是一个好的拟合。

- 不同疾病患病人数柱状图

我们可以分析不同疾病患病人数之间的差异。对于这种问题我们最好的选择是绘制柱状图，他是一种以长方形的长度为变量的统计图表。长条图用来比较两个或以上的价值（不同时间或者不同条件），只有一个变量，通常利用于较小的数据集分析。[11]而患病人数是这张图表中的唯一变量，且之间不存在随时间变化的关系，即在一段时间内的患病人数，因此适用于使用柱状图。分析所使用的代码如下：

```
# 9.不同疾病患病人数的柱状图（按性别分组）

disease_gender_counts = data.groupby(['Medical Condition',
'Gender']).size().unstack().fillna(0)

disease_gender_counts['Total'] = disease_gender_counts.sum(axis=1)

disease_gender_counts = disease_gender_counts.sort_values(by='Total',
ascending=False).drop(columns='Total')
```

```
plt.figure(figsize=(14, 6))

bar_width = 0.4

index = np.arange(len(disease_gender_counts))

plt.bar(index - bar_width/2, disease_gender_counts['Male'], width=bar_width, label='男')

plt.bar(index + bar_width/2, disease_gender_counts['Female'], width=bar_width, label='女')

plt.xlabel('疾病名称')

plt.ylabel('患病人数')

plt.title('不同疾病患病人数统计（按性别）')

plt.xticks(index, disease_gender_counts.index, rotation=45, ha='right')

plt.legend()

plt.tight_layout()

plt.savefig("不同疾病患病人数统计_柱状图_按性别.png")
```

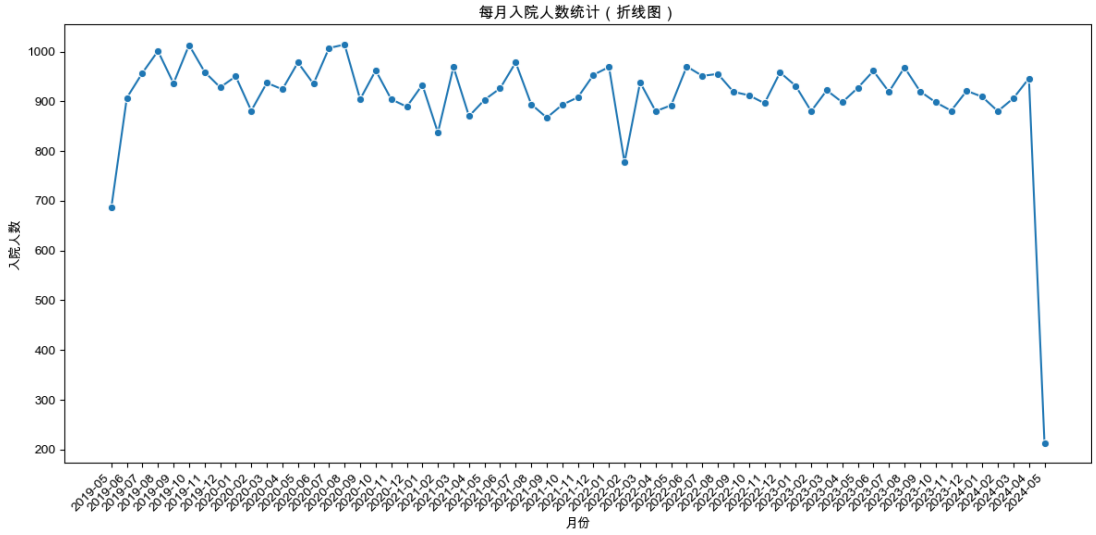


通过图表可以观察到，各疾病之间不存在显著的性别差异，患病人数也不存在显著的差异。

- 患病人数与实践之间的关系

```
data['Date of Admission'] = pd.to_datetime(data['Date of Admission'], errors='coerce')

data_valid_dates = data.dropna(subset=['Date of Admission'])

data_valid_dates['Month'] = data_valid_dates['Date of Admission'].dt.to_period('M').astype(str)
```

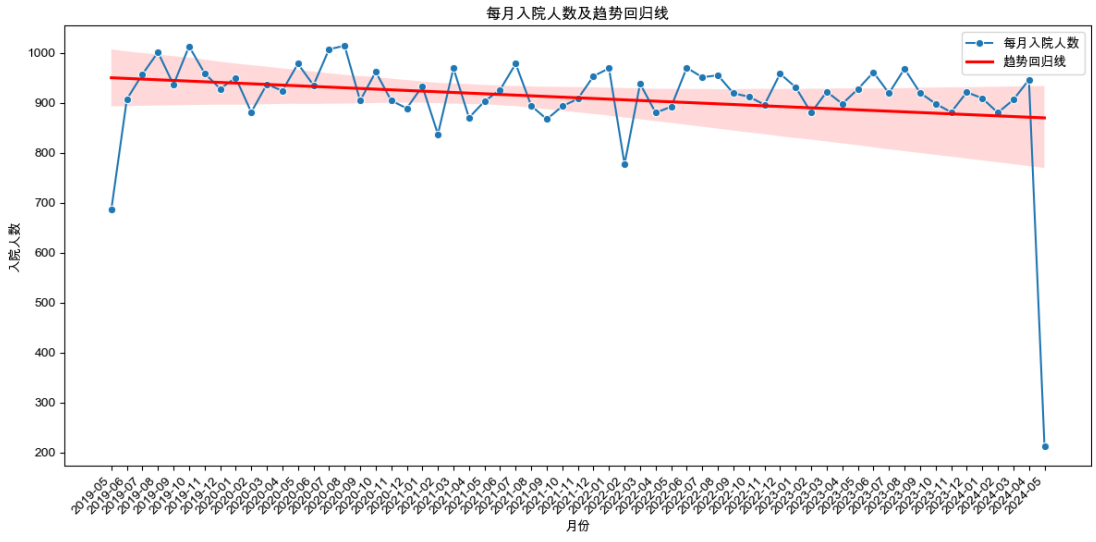
```
monthly_counts = data_valid_dates['Month'].value_counts().sort_index()
plt.figure(figsize=(12, 6))
sns.lineplot(x=monthly_counts.index, y=monthly_counts.values, marker='o')
plt.title("每月入院人数统计（折线图）")
plt.xlabel("月份")
plt.ylabel("入院人数")
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.savefig("每月入院人数折线图.png")
```



通过这张图片我们发现入院人数似乎随着时间的推移而逐渐减少，因此我们修改代码，在这张图上继续画出入院人数和时间的回归方程。

```
# 10.每月入院人数折线图（按 Date of Admission 统计）
data['Date of Admission'] = pd.to_datetime(data['Date of Admission'], errors='coerce')
data_valid_dates = data.dropna(subset=['Date of Admission'])
data_valid_dates['Month'] = data_valid_dates['Date of Admission'].dt.to_period('M').astype(str)
monthly_counts = data_valid_dates['Month'].value_counts().sort_index()
monthly_df = pd.DataFrame({
    'Month': monthly_counts.index,
```

```
    'Count': monthly_counts.values
})
monthly_df['Month_Num'] = np.arange(len(monthly_df))
plt.figure(figsize=(12, 6))
sns.lineplot(x='Month_Num', y='Count', data=monthly_df, marker='o', label='每月入院人数')
sns.regplot(x='Month_Num', y='Count', data=monthly_df, scatter=False, color='red', label='趋势回归线')
plt.xticks(ticks=monthly_df['Month_Num'], labels=monthly_df['Month'], rotation=45, ha='right')
plt.xlabel("月份")
plt.ylabel("入院人数")
plt.title("每月入院人数及趋势回归线")
plt.legend()
plt.tight_layout()
plt.savefig("10 每月入院人数_含趋势线.png")
```



绘制的结果证实了猜想。随着时间的推移每月的入院人数有下降趋势，这可能是由多方面因素共同作用的结果。可能是地区人口减少、医疗水平的提高、居民健康情况改善、地区环境治理成效、医疗保险政策调整等多方面的。这当然是一个好的趋势，但如果我们需要分析具体原因，需要更多数据集的支持。

经过了上述 10 段数据分析，我们对医院在这段时间内对数据情况有了大致的了解。接下来我们想使用 scikit-learn 构建相应的模型，根据其他的数据来预测患各种疾病人数，帮助医院合理安排医疗资源。

- 对数据集进行划分，20%用于测试集，80%为训练集，使用支持向量机模型 SVM 构建分类模型。所使用的代码如下：

```
features = ['Hospital', 'Gender', 'Age', 'Length of Stay', 'Insurance Provider',
                    'Test Results', 'Medication']
target = 'Medical Condition'
X = pd.get_dummies(data_cleaned[features], drop_first=True)
y = data_cleaned[target]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
svm_model = SVC(kernel='rbf', C=1.0, gamma='scale', random_state=42)
svm_model.fit(X_train_scaled, y_train)
y_pred = svm_model.predict(X_test_scaled)
print("分类报告：")
print(classification_report(y_test, y_pred))
print("混淆矩阵：")
cm=confusion_matrix(y_test, y_pred)
print(cm)
plt.figure(figsize=(10, 8))
labels = svm_model.classes_
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=labels,
```

yticklabels=labels)

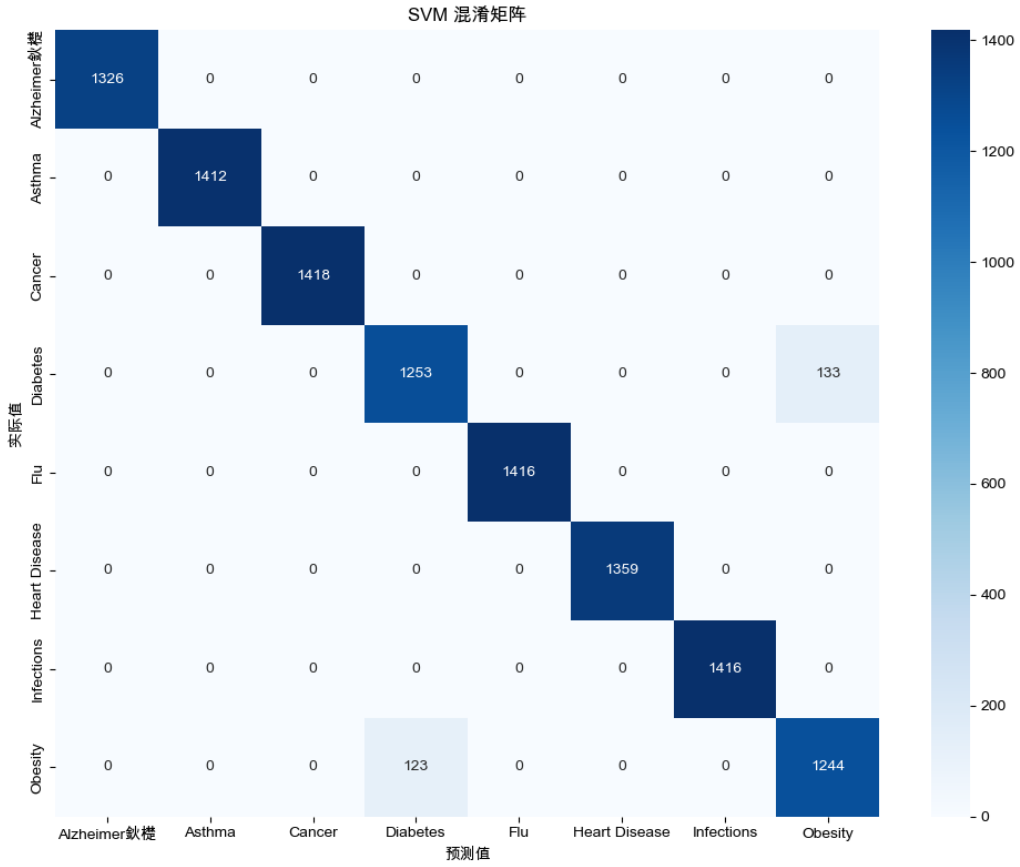
plt.xlabel("预测值")

plt.ylabel("实际值")

plt.title("SVM 混淆矩阵")

plt.tight_layout()

plt.savefig("12 混淆矩阵.png")



上述混淆矩阵表明，我们使用数据集中的'Hospital', 'Gender', 'Age', 'Length of Stay', 'Insurance Provider','Test Results', 'Medication'关键字对 Medical Condition 的预测准确率极高，仅有少部分 Diabetes 和 Obesity 问题无法正确判断。

- SVM 支持向量机的评价

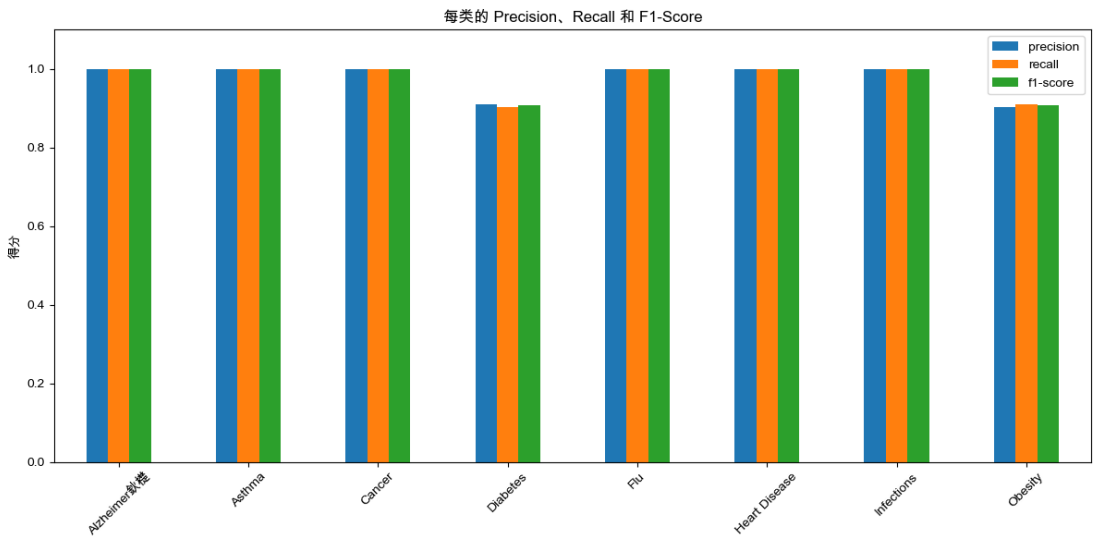
这里我们使用的代码如下：

print(classification_report(y_test, y_pred))

kappa = cohen_kappa_score(y_test, y_pred)

print(f"\nCohen's Kappa 系数：{kappa:.4f}")

```
report_dict = classification_report(y_test, y_pred, output_dict=True)
report_df = pd.DataFrame(report_dict).transpose()
report_df = report_df.loc[report_df.index[:-3]]
report_df[['precision', 'recall', 'f1-score']].plot(kind='bar', figsize=(12, 6))
plt.title('每类的 Precision、Recall 和 F1-Score')
plt.ylabel('得分')
plt.ylim(0, 1.1)
plt.xticks(rotation=45)
plt.tight_layout()
plt.savefig('13svm 分类模型的评价.png')
```



于是我们得到了如下分类性能报告：

	precision	recall	f1-score	support
Alzheimer 鈥欑	1.00	1.00	1.00	1326
Asthma	1.00	1.00	1.00	1412
Cancer	1.00	1.00	1.00	1418
Diabetes	0.91	0.90	0.91	1386
Flu	1.00	1.00	1.00	1416
Heart Disease	1.00	1.00	1.00	1359
Infections	1.00	1.00	1.00	1416
Obesity	0.90	0.91	0.91	1367
accuracy			0.98	11100
Macro avg	0.98	0.98	0.98	11100

Weighted avg	0.98	0.98	0.98	11100

总体准确率（Accuracy）：98%，加权平均 F1-score：0.98，Cohen's Kappa 系数：0.9736，表明模型预测与真实标签之间具有极高的一致性，远高于随机分类水平。

对于大多数疾病类别，模型实现了完美分类（F1=1.00），表现极为优秀。对于糖尿病与肥胖两类，虽然 F1 略低于 1，但仍处于较高水平（0.91），该 SVM 支持向量机模型对这些边界样本的处理仍具备很好的区分能力。

5 实验结果与分析

用 Python，把数据集中的不同属性用图展示出来并进行简单剖析，体现了 Python 作为一门高级语言在现代化数据分析中所起到的作用。

本次试验中我使用了从 Kaggle 上下载的关于医疗健康的数据集，使用 Pandas+Seaborn+Matplotlib+Sklearn 对这个 CSV 文件进行整合、加工。数据集表示不同的保险公司和不同的医院在接诊患者的过程中对医疗费用没有显著影响，而疾病诊断的难易程度和疾病的类型对医疗行为所产生的账单费用影响最大。通过对数据模型的回归分析我们发现数据集展现的居民健康状况随着时间的推移正在逐步好转，这可能是由多种因素造成的，而好转的原因的研究需要更多数据集的支持。通过构建 SVM 支持向量机模型，我们可以通过其他的几项参数对疾病的种类进行分类，并且获得了较高的准确性，足以体现此 SVM 模型的准确性。

未来的研究方向。未来，可以探索更多的算法、技术手段，使用跟先进的算法和技术手段来分析疾病的种类和治疗账单多少的影响因素。可以继续收集该地区生活习惯、环境污染等更多的资料，通过全方位多层次的数据分析，寻找到影响当地居民身体健康的可能因素。

参考文献

[1] Eduardo Licea. Healthcare Dataset[EB/OL]. [2024-04-20]. https://www.kaggle.com/datasets/eduardolicea/healthcare-dataset/data.

[2] 维基百科编者. Pandas[G/OL]. 维基百科, 2025(20250317)[2025-04-20]. https://zh.wikipedia.org/w/inex.php?title=Pandas&oldid=86470885

[3] 曾文权，张良均主编，黄红梅，施兴，黄添喜副主编. Python 数据分析与应用[M]. 第 2 版. 北京：中国工信出版集团，人民邮电出版社，2021：54.

[4] deephub. CSDN2020(20200818)[2025-04-20]. https://blog.csdn.net/deephub/article/details/108068984#:~:text=Seaborn%20%E6%98%AF%E4%B8%80%E4%B8%AA%E5%9F%BA%E4%BA%8Ematplotlib,%E8%81%9A%E5%90%88%E4%BB%A5%E7%94%9F%E6%88%90%E4%BF%A1%E6%81%AF%E5%9B%BE%E3%80%82

[5] 中国科学院大学本科部，2025(20250320)[2025-04-25]，相关系数，百度百科 https://baike.baidu.com/item/%E7%9B%B8%E5%85%B3%E7%B3%BB%E6%95%B0/3109424

[6] 维基百科编者. 协方差[G/OL]. 维基百科, 2023(20231121)[2023-11-21]. https://zh.wikipedia.org/w/index.php?title=%E5%8D%8F%E6%96%B9%E5%B7%AE&oldid=79825035.

[7] 维基百科编者. 分位数[G/OL]. 维基百科, 2021(20211117)[2021-11-17]. https://zh.wikipedia.org/w/index.php?title=%E5%88%86%E4%BD%8D%E6%95%B0&oldid=68701010.

[8] 沈海军，2025(20231229)[2025-05-04]，独立性检验，百度百科 https://baike.baidu.com/item/%E7%8B%AC%E7%AB%8B%E6%80%A7%E6%A3%80%E9%AA%8C/4031921

[9] 维基百科编者. 支持向量机[G/OL]. 维基百科, 2025(20250203)[2025-02-03]. https://zh.wikipedia.org/w/index.php?title=%E6%94%AF%E6%8C%81%E5%90%91%E9%87%8F%E6%9C%BA&oldid=85948541.

[10] 徐豪,刘婉月,张自豪.基于 Pandas+Seaborn+Matplotlib 的城市共享单车租赁分析可视化[J].现代信息科技,2024,8(23):58-62+68.DOI:10.19850/j.cnki.2096-4706.2024.23.013.

[11] 维基百科编者. 条形图[G/OL]. 维基百科, 2024(20240912)[2024-09-12]. https://zh.wikipedia.org/w/index.php?title=%E6%9D%A1%E5%BD%A2%E5%9B%BE&oldid=84180253.