任务2.1

X3 x4

回答问题

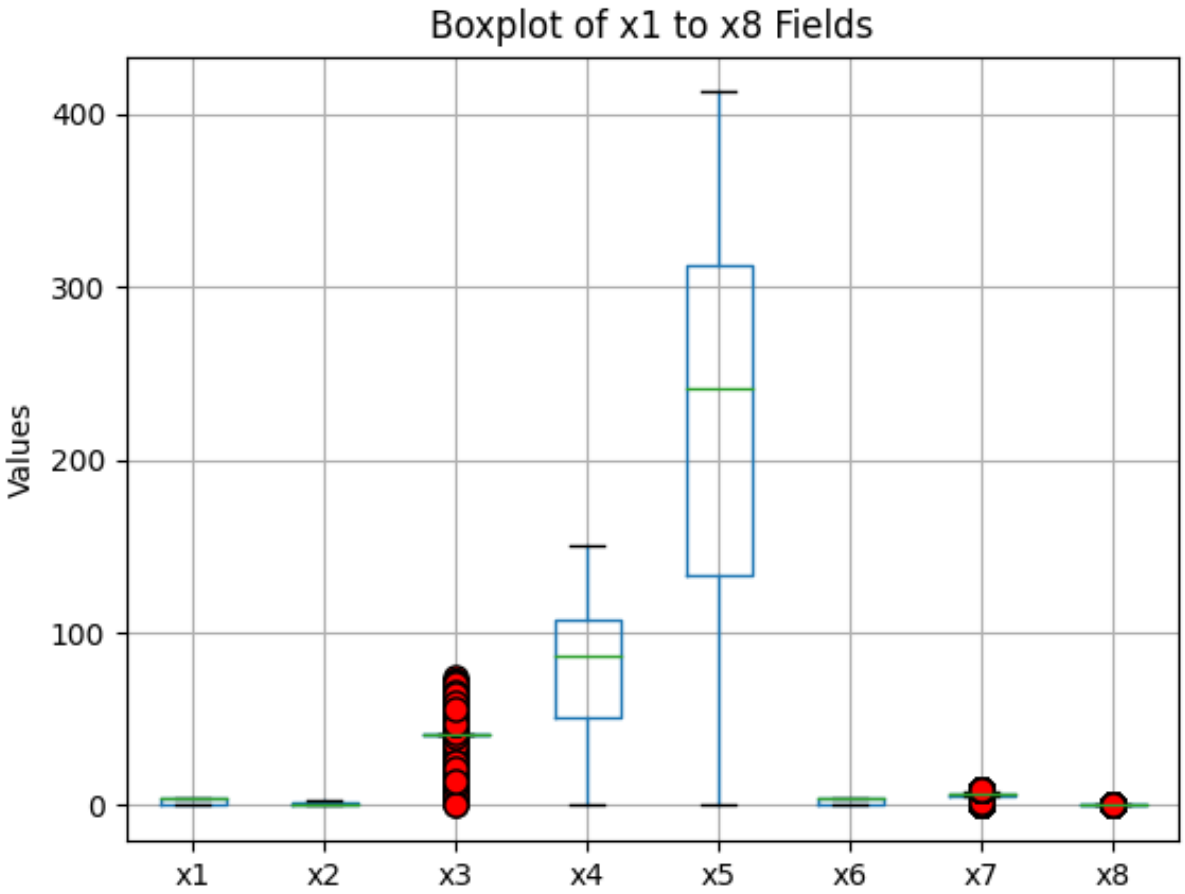
- 字段x1至x8为用户相关的属性，为匿名处理字段。添加代码对这些数据字段的取值分析，那些字段为数值类型？那些字段为类别类型?

```
def find_x():
    train_data = pd.read_csv('./data/train_new.csv')
    test_data = pd.read_csv('./data/test_new.csv')

    # 假设这是数据集中的字段名列表
    # field_names = ['uuid', 'eid', 'udmap', 'common_ts', 'x1', 'x2', 'x3', 'x4', 'x5',
'x6', 'x7', 'x8', 'target',
    #                'key1', 'key2', 'key3', 'key4', 'key5', 'key6', 'key7', 'key8',
'key9', 'udmap_isunknown',
    #                'eid_freq', 'eid_mean', 'common_ts_hour']
    field_names = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']

    # 初始化一个字典用于存储字段类型信息
    field_types = {}

    # 遍历每个字段，分析其取值类型
    for field_index, field_name in enumerate(field_names):
        # 假设采用简单的规则：如果所有样本都可以转换为浮点数，则为数值类型；否则为类别类型
        is_numeric = all(isinstance(value, (int, float)) for value in
train_data[field_name])
        field_types[field_name] = 'Numeric' if is_numeric else 'Categorical'

    # 打印字段类型信息
    for field_name, field_type in field_types.items():
        print(f"Field '{field_name}' is of type: {field_type}")
```

结果都是数字类型。train和test都是数值类型

- 对于数值类型的字段，考虑绘制在标签分组下的箱线图。

```python
def print_viobox():
    # 绘制箱线图
    train_data = pd.read_csv('./data/train_new.csv')
    test_data = pd.read_csv('./data/test_new.csv')
    field_names = ['x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8']

    test_data[field_names].boxplot()
    # 设置标题和标签
    plt.title('Boxplot of x1 to x8 Fields')
    plt.ylabel('Values')
    plt.show()
```



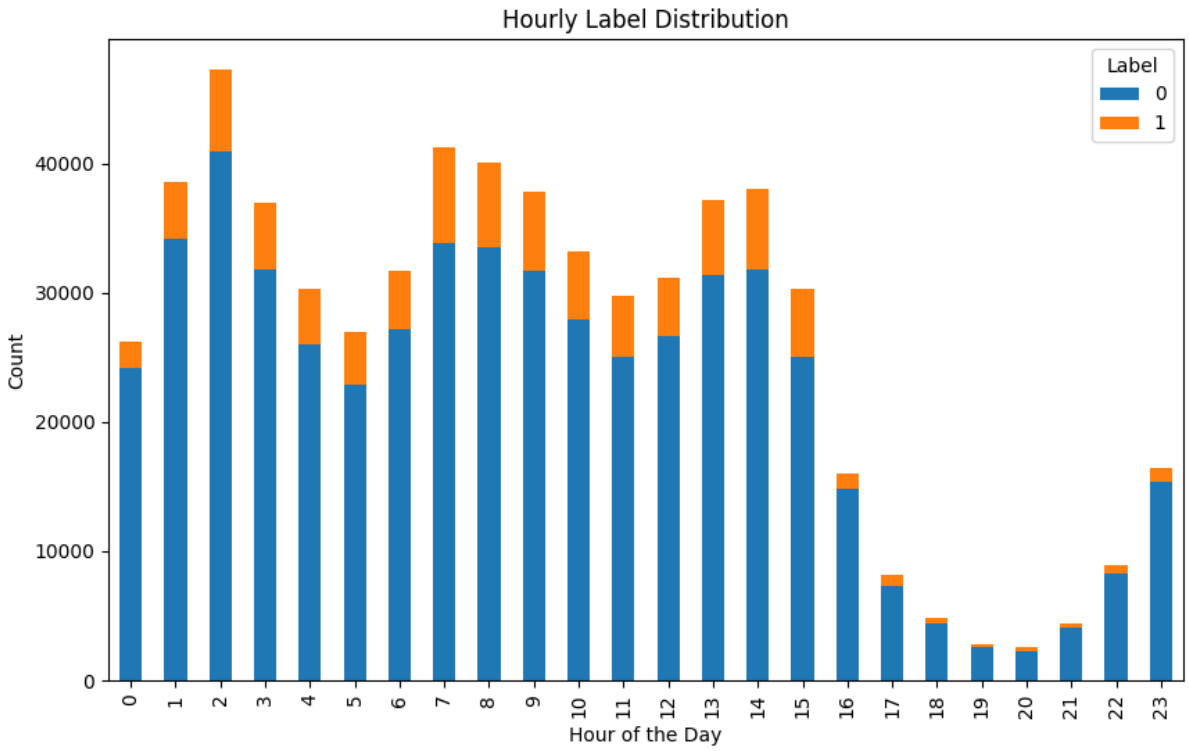
看出x1,x2,x6集中于0,3 7 8 是由很多异常值

```
/usr/bin/python3 /Users/shawnzhao/Code/User_Predict/main.py
                  x3             x7             x8
count  620356.000000  620356.000000  620356.000000
mean       40.974499       5.863720       0.855459
std         1.373016       2.575854       0.351638
min         0.000000       0.000000       0.000000
25%        41.000000       6.000000       1.000000
50%        41.000000       7.000000       1.000000
75%        41.000000       7.000000       1.000000
max        74.000000       9.000000       1.000000
Number of records where x3 is not 41: 2503
Number of records where x7 is not 7: 435084
Number of records where x8 is not 1: 89667
```

看的出，x3集中于41，x7集中于7 x8集中于0。

- 从**common_ts**中提取小时，绘制每小时下标签分布的变化。

```
def hour_change():
    train_data = pd.read_csv('./data/train_new.csv')
    # 统计每小时下标签的分布
    hourly_label_counts = train_data.groupby('common_ts_hour')
['target'].value_counts().unstack().fillna(0)

    # 绘制每小时下标签分布的变化
    hourly_label_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
    plt.title("Hourly Label Distribution")
    plt.xlabel("Hour of the Day")
    plt.ylabel("Count")
    plt.legend(title='Label')
    plt.show()
```



- 对**udmap**进行onehot，统计每个key对应的标签均值，绘制直方图。

任务2.2

```
# 训练并验证SGDClassifier
    pred = cross_val_predict(
        SGDClassifier(max_iter=10),
        train_data.drop(['udmap', 'common_ts', 'uuid', 'target'], axis=1),
        train_data['target']
    )
    print(classification_report(train_data['target'], pred, digits=3))

#################
              precision    recall  f1-score   support

           0      0.868     0.871     0.869    533155
           1      0.194     0.191     0.193     87201

    accuracy                          0.775    620356
   macro avg      0.531     0.531     0.531    620356
weighted avg      0.773     0.775     0.774    620356
##############################
# 训练并验证DecisionTreeClassifier
    pred = cross_val_predict(
        DecisionTreeClassifier(),
        train_data.drop(['udmap', 'common_ts', 'uuid', 'target'], axis=1),
        train_data['target']
```

```
    )
    print(classification_report(train_data['target'], pred, digits=3))
###################
              precision    recall  f1-score   support

           0      0.934     0.940     0.937    533155
           1      0.618     0.592     0.605     87201

    accuracy                          0.891    620356
   macro avg      0.776     0.766     0.771    620356
weighted avg      0.889     0.891     0.890    620356
################################
# 训练并验证MultinomialNB 朴素贝叶斯
    pred = cross_val_predict(
        MultinomialNB(),
        train_data.drop(['udmap', 'common_ts', 'uuid', 'target'], axis=1),
        train_data['target']
    )
    print(classification_report(train_data['target'], pred, digits=3))
###################
              precision    recall  f1-score   support

           0      0.893     0.736     0.807    533155
           1      0.221     0.458     0.298     87201

    accuracy                          0.697    620356
   macro avg      0.557     0.597     0.552    620356
weighted avg      0.798     0.697     0.735    620356
################################
# 训练并验证RandomForestClassifier
    # 训练并验证RandomForestClassifier
pred = cross_val_predict(
    RandomForestClassifier(n_estimators=91),
    train_data.drop(['udmap', 'common_ts', 'uuid', 'target'], axis=1),
    train_data['target']
)
print(classification_report(train_data['target'], pred, digits=10))
################
                     precision    recall  f1-score   support

           0  0.9219746475 0.9674841275 0.9441813171    533155
           1  0.7152618093 0.4993979427 0.5881486984     87201

    accuracy                      0.9016870958    620356
   macro avg  0.8186182284 0.7334410351 0.7661650078    620356
weighted avg  0.8929178379 0.9016870958 0.8941352140    620356
```

`precision`, `recall`, `f1-score` 和 `support` 是分类报告中的重要指标。

- `precision`（精确度）：预测为正类别的样本中实际为正类别的比例。
- `recall`（召回率）：实际为正类别的样本中被正确预测为正类别的比例。
- `f1-score`（F1 分数）：精确度和召回率的调和平均值，用于综合考虑分类器的性能。
- `support`（支持数）：每个类别的实际样本数。

这些指标提供了关于分类器性能的全面评估。通常，人们会根据具体的问题情况，选择合适的指标来进行模型的评估和选择。

- `accuracy`（准确率）：正确预测的样本数与总样本数之比，是分类正确率的度量。
- `macro avg`（宏平均）：对各个类别指标的简单平均值，不考虑样本不平衡。
- `weighted avg`（加权平均）：对各个类别指标的加权平均值，考虑样本不平衡。

这些指标都可以帮助你更好地理解分类器的性能。总体来说，`accuracy` 是分类器整体的准确率，`macro avg` 和 `weighted avg` 则是综合考虑各个类别的指标。在评估分类器性能时，需要综合考虑这些指标以及具体的业务需求。

```
### 首先明确一下几个表示： -True Positive(真正, TP)：将正类预测为正类数. -True Negative(真负, TN)：将负类预测为负类数. -False Positive(假正, FP)：将负类预测为正类数 →→ 误报 (Type I error). -False Negative(假负, FN)：将正类预测为负类数 →→ 漏报 (Type II error).

### 精确率：precision = TP / (TP + FP)

### 召回率：recall = TP / (TP + FN)

### 准确率：accuracy = (TP + TN) / (TP+ FP + TN + FN)
精确率是针对我们预测结果而言的，它表示的是预测为正的样本中有多少是真正的正样本。召回率是针对我们原来的样本而言的，它表示的是样本中的正例有多少被预测正确了。其实就是分母不同，一个分母是预测为正的样本数，另一个是原来样本中所有的正样本数。
```

F1 分数的计算公式如下：

$$F1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1 分数的取值范围在 0 到 1 之间，越接近 1 表示分类器的性能越好，同时兼顾了精确度和召回率。在某些情况下，你可能更关心精确度，而在另一些情况下可能更关心召回率。F1 分数能够帮助你在精确度和召回率之间找到一个平衡点，从而更好地评估分类器的表现。