```
# (1)intro:数据导入
```

```
import pandas as pd

# 字段配置
TIME_COL = 'tenure'
EVENT_COL = 'Churn'
DROP_COLS = ['customerID']
CAT_COLS = [
    'gender', 'Partner', 'Dependents',
    'PhoneService', 'MultipleLines',
    'InternetService', 'OnlineSecurity',
    'OnlineBackup', 'DeviceProtection',
    'TechSupport', 'StreamingTV',
    'StreamingMovies', 'Contract',
    'PaperlessBilling', 'PaymentMethod'
]

# === 数据加载和处理 ===
# 加载数据
df = pd.read_csv("data1.csv", na_values=[' '])

# 转换字段类型
df['TotalCharges'] = pd.to_numeric(df['TotalCharges'], errors='coerce')
df[EVENT_COL] = df[EVENT_COL].map({'Yes': 1, 'No': 0})

# 月度合同用户
df = df[df['Contract'] == 'Month-to-month']

# 关键字段
keep_cols = [TIME_COL, EVENT_COL] + CAT_COLS
df = df[keep_cols].drop(columns=DROP_COLS, errors='ignore')

# 缺失值处理
df[TIME_COL] = df[TIME_COL].fillna(0)
df = df.dropna(subset=[EVENT_COL])

# 校验数据合法性
assert df[TIME_COL].min() >= 0, "存在负时间值"
assert set(df[EVENT_COL]) <= {0, 1}, "事件列包含非法值"

# === 快速报告 ===
report = {
    "样本量": len(df),
    "事件发生率": f"{df[EVENT_COL].mean():.1%}",
    "平均持续时间": f"{df[TIME_COL].mean():.1f}月",
    "特征维度": df.shape[1]
}

print("数据状态简报:")
print(pd.Series(report))

print("\n数据结构示例:")
print(df.iloc[:3, :5])  # 显示前3行和前5列
```

```
数据状态简报:
样本量          3875
事件发生率       42.7%
平均持续时间     18.0月
特征维度          17
dtype: object

数据结构示例:
   tenure  Churn  gender Partner Dependents
0       1      0  Female     Yes         No
2       2      1    Male      No         No
4       2      1  Female      No         No
```

```
# (2)kaplan_meier
```

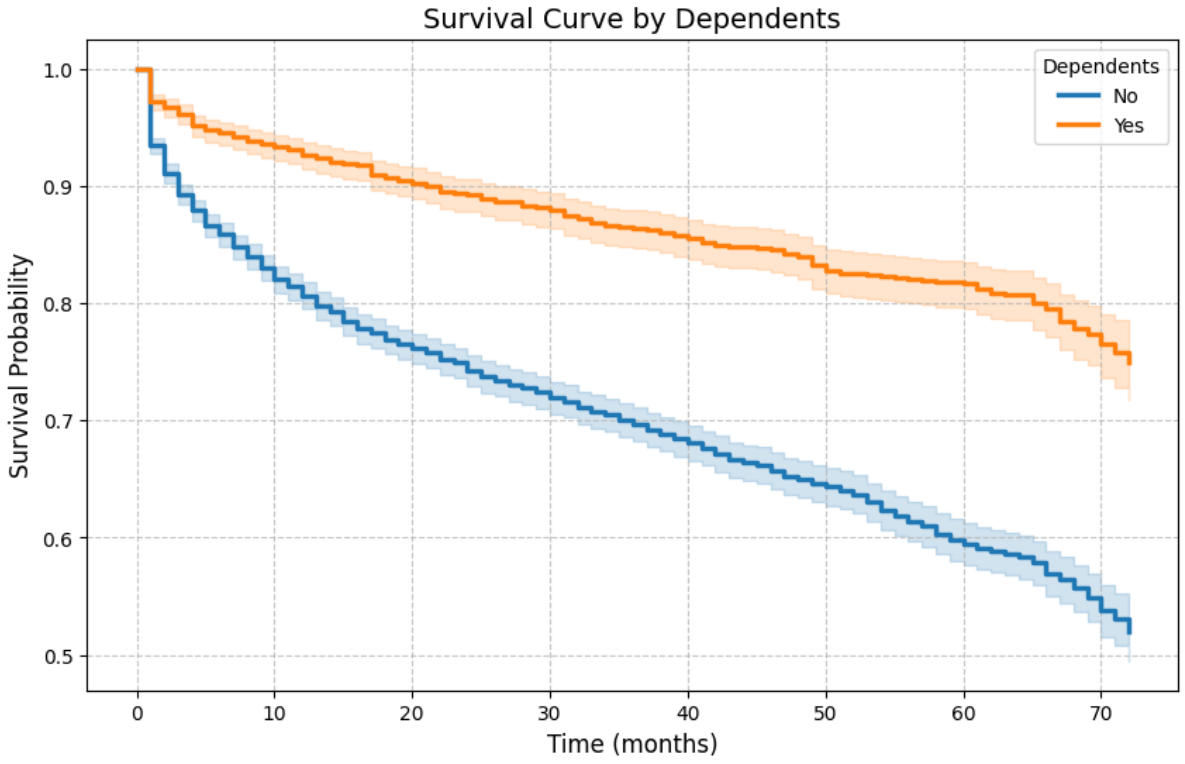
```
"""
功能：生存曲线绘制、Log-rank检验、结果导出
"""
import pandas as pd
import matplotlib.pyplot as plt
from lifelines import KaplanMeierFitter
from lifelines.statistics import pairwise_logrank_test
import os

# 全局样式配置
PLOT_STYLE = {
    'figsize': (10, 6),
    'title_fontsize': 14,
    'label_fontsize': 12,
    'colors': ['#1f77b4', '#ff7f0e', '#2ca02c', '#d62728',
               '#9467bd', '#8c564b', '#e377c2', '#7f7f7f'],
    'ci_alpha': 0.2,
    'line_styles': ['-', '--', '-.', ':'],
    'line_width': 2.5
}

def validate_data(df, duration_col, event_col):
    """数据验证"""
    required_cols = [duration_col, event_col]
    missing = [col for col in required_cols if col not in df.columns]
    if missing:
        raise ValueError(f"缺失必要列: {missing}")
    if df[event_col].nunique() != 2:
        raise ValueError("事件列必须为二分类变量 (0/1)")

def fit_models(df, duration_col, event_col, group_col=None):
    """拟合生存模型"""
    if group_col is None:
        kmf = KaplanMeierFitter()
        kmf.fit(df[duration_col], df[event_col])
```

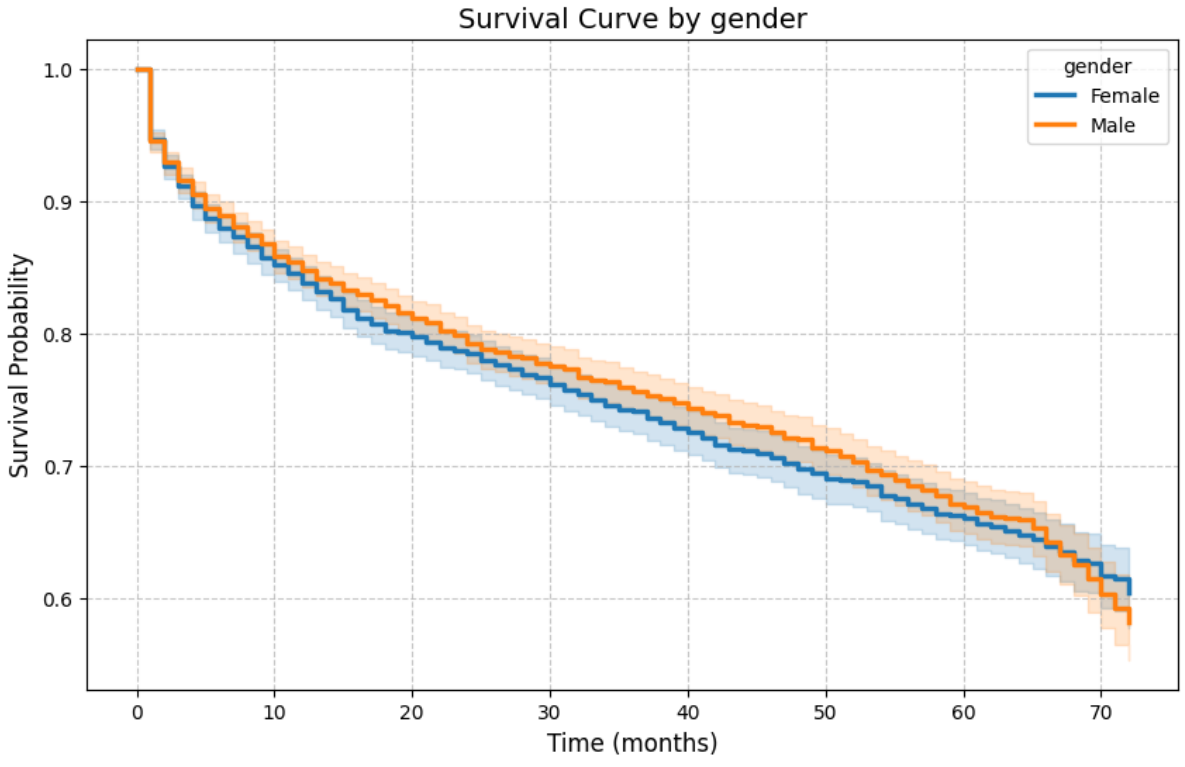
```
        return {'Overall': kmf}
    else:
        groups = df[group_col].unique()
        return {
            group: KaplanMeierFitter().fit(
                df[df[group_col]==group][duration_col],
                df[df[group_col]==group][event_col]
            )
            for group in groups
        }

def plot_curves(df, duration_col, event_col, group_col=None, save_path=None):
    """绘制生存曲线"""
    plt.figure(figsize=PLOT_STYLE['figsize'])
    ax = plt.gca()

    models = fit_models(df, duration_col, event_col, group_col)

    for i, (group_name, model) in enumerate(models.items()):
        color_idx = i % len(PLOT_STYLE['colors'])
        line_idx = i // len(PLOT_STYLE['colors'])

        model.plot_survival_function(
            ax=ax,
            label=group_name if group_col else None,
            color=PLOT_STYLE['colors'][color_idx],
            linestyle=PLOT_STYLE['line_styles'][line_idx],
            linewidth=PLOT_STYLE['line_width'],
            ci_alpha=PLOT_STYLE['ci_alpha']
        )

    title = "Survival Curve" + (f" by {group_col}" if group_col else "")
    ax.set_title(title, fontsize=PLOT_STYLE['title_fontsize'])
    ax.set_xlabel("Time (months)", fontsize=PLOT_STYLE['label_fontsize'])
    ax.set_ylabel("Survival Probability", fontsize=PLOT_STYLE['label_fontsize'])
    ax.grid(True, linestyle='--', alpha=0.7)

    if group_col:
        plt.legend(title=group_col)


    os.makedirs(os.path.dirname(save_path), exist_ok=True)
    plt.savefig(save_path, bbox_inches='tight')
    plt.close()
    plt.show()

def logrank_test(df, duration_col, event_col, group_col):
    """执行Log-rank检验"""
    if df[group_col].nunique() < 2:
        raise ValueError("分组列至少需要两个类别")

    results = pairwise_logrank_test(
        event_durations=df[duration_col],
        groups=df[group_col],
        event_observed=df[event_col]
    )
```

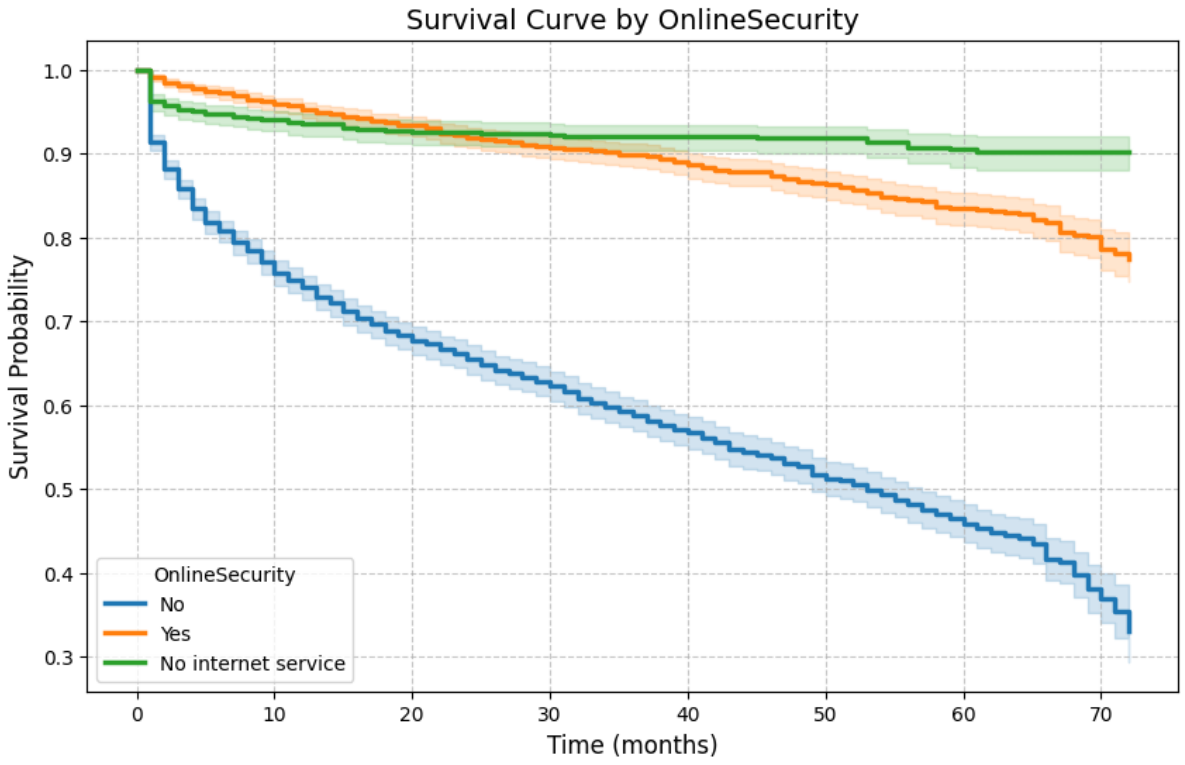
```python
    return results.summary

def analyze_feature(df, duration_col, event_col, group_col, output_dir=None):
    """完整分析流程"""
    # 数据验证
    validate_data(df, duration_col, event_col)

    # 绘制生存曲线
    save_path = f"{output_dir}/km_{group_col}.png" if output_dir else None
    plot_curves(df, duration_col, event_col, group_col, save_path)

    # 执行统计检验
    test_result = logrank_test(df, duration_col, event_col, group_col)
    print(f"\nLog-rank检验结果 ({group_col}):")
    print(test_result)

    return test_result

if __name__ == "__main__":
    # 示例数据准备
    data = pd.read_csv("data1.csv")
    data['Churn'] = data['Churn'].map({'Yes': 1, 'No': 0})

    # 分析单个特征
    analyze_feature(data, 'tenure', 'Churn', 'OnlineSecurity',
output_dir="./results")

    # 批量分析多个特征
    features = ['gender', 'Partner', 'Dependents', 'OnlineSecurity']
    for feature in features:
        analyze_feature(data, 'tenure', 'Churn', feature, output_dir="./results")
```

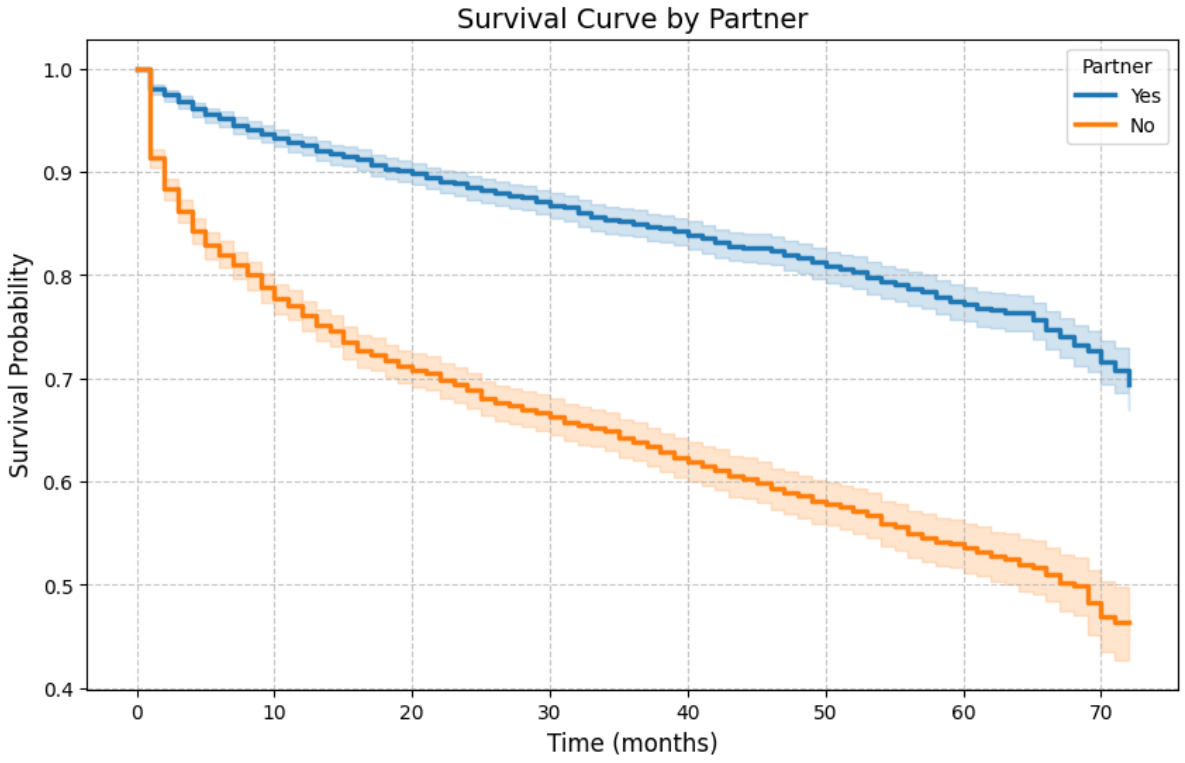
```
Log-rank检验结果 (OnlineSecurity):
                                      test_statistic              p  \
No                 No internet service      485.975805  1.070135e-107
                   Yes                      660.525069  1.148535e-145
No internet service Yes                      11.060731   8.817539e-04

                                        -log2(p)
No                 No internet service  355.348514
                   Yes                  481.479779
No internet service Yes                  10.147336

Log-rank检验结果 (gender):
             test_statistic         p  -log2(p)
Female Male        0.525707  0.468417  1.094134

Log-rank检验结果 (Partner):
        test_statistic             p    -log2(p)
No Yes      423.543082  4.132951e-94  310.214069

Log-rank检验结果 (Dependents):
        test_statistic             p   -log2(p)
No Yes      232.699042  1.537238e-52  172.11992
```

```
Log-rank检验结果 (OnlineSecurity):
                                        test_statistic             p  \
No                  No internet service      485.975805  1.070135e-107
                    Yes                      660.525069  1.148535e-145
No internet service Yes                       11.060731   8.817539e-04

                                          -log2(p)
No                  No internet service  355.348514
                    Yes                  481.479779
No internet service Yes                   10.147336
```










```
# (3)cox_proportional_hazards
```

```
"""
Cox比例风险模型分析
功能：数据预处理、模型拟合、假设检验、结果可视化
"""
import os
import pandas as pd
import matplotlib.pyplot as plt
from lifelines import CoxPHFitter
```

```python
# === 配置区 ===
DATA_PATH = "data1.csv"
DURATION_COL = 'tenure'
EVENT_COL = 'Churn'
CAT_COLS = ['gender', 'InternetService', 'Contract', 'PaymentMethod']
DROP_REF = True
INCLUDE_COLS = [
    'Gender_Male',
    'internetservice_fiber_optic',
    'contract_one_year'
]

# === 工具函数 ===
def clean_col_name(name):
    return name.strip().lower().replace(" ", "_").replace("-", "_")

# === 加载并验证数据 ===
df = pd.read_csv(DATA_PATH)
print("原始数据列名:", df.columns.tolist())

if DURATION_COL not in df.columns or EVENT_COL not in df.columns:
    raise KeyError(f"列名错误: '{DURATION_COL}' 或 '{EVENT_COL}' 不存在于数据中。")

# === 预处理 ===
duration_event_df = df[[DURATION_COL, EVENT_COL]].copy()

# 分类变量 one-hot 编码
if CAT_COLS:
    encoded = pd.get_dummies(df[CAT_COLS], prefix_sep='_', drop_first=DROP_REF)
    encoded.columns = [clean_col_name(c) for c in encoded.columns]
    analysis_df = pd.concat([duration_event_df, encoded], axis=1)
else:
    analysis_df = duration_event_df.copy()

# 事件列格式统一
analysis_df[EVENT_COL] = (
    analysis_df[EVENT_COL]
    .replace({'Yes': 1, 'No': 0, 'yes': 1, 'no': 0})
    .astype(float)
)

# === 特征选择 ===
feature_cols = [
    col for col in analysis_df.columns
    if col not in [DURATION_COL, EVENT_COL]
]

if INCLUDE_COLS:
    include_cols_clean = [clean_col_name(c) for c in INCLUDE_COLS]
    normalized_features = [clean_col_name(c) for c in feature_cols]
    selected = [
        feature_cols[normalized_features.index(c)]
        for c in include_cols_clean
        if c in normalized_features
    ]
```

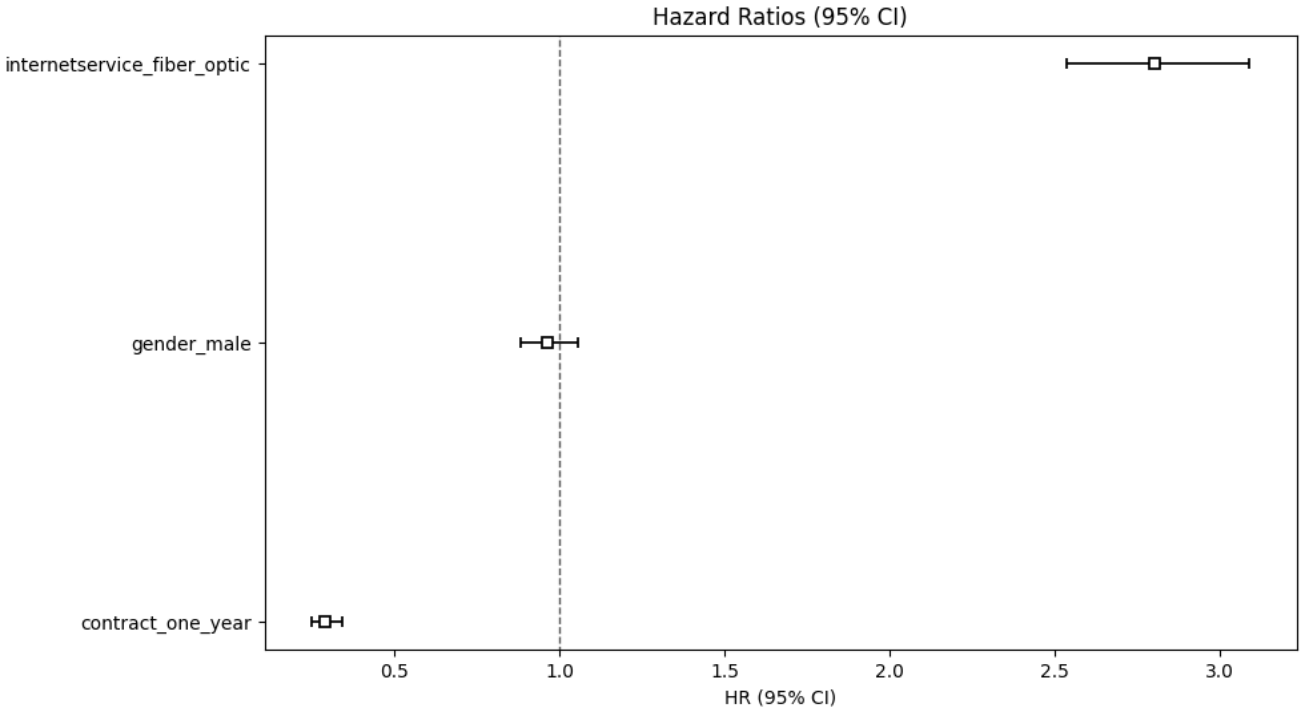
```
else:
    selected = feature_cols

print("\n参与建模的特征列:")
print(selected)

# === 拟合Cox模型 ===
cph = CoxPHFitter()
model_data = analysis_df[[DURATION_COL, EVENT_COL] + selected]
cph.fit(model_data, duration_col=DURATION_COL, event_col=EVENT_COL)

# === 输出模型报告 ===
print("\n模型摘要:")
cph.print_summary()

# === 可视化HR ===
plt.figure(figsize=(10, 6))
cph.plot(hazard_ratios=True)
plt.title("Hazard Ratios (95% CI)")
os.makedirs("./cox_results", exist_ok=True)
plt.savefig("./cox_results/hazard_ratios.png")
plt.show()
```

```
原始数据列名: ['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
'tenure', 'PhoneService', 'MultipleLines', 'InternetService', 'OnlineSecurity',
'OnlineBackup', 'DeviceProtection', 'TechSupport', 'StreamingTV',
'StreamingMovies', 'Contract', 'PaperlessBilling', 'PaymentMethod',
'MonthlyCharges', 'TotalCharges', 'Churn']

参与建模的特征列:
['gender_male', 'internetservice_fiber_optic', 'contract_one_year']
```

model	lifelines.CoxPHFitter
duration col	'tenure'
event col	'Churn'
baseline estimation	breslow
number of observations	7043
number of events observed	1869
partial log-likelihood	-15255.27
time fit was run	2025-04-13 13:13:57 UTC

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

Concordance	0.68
Partial AIC	30516.53
log-likelihood ratio test	795.55 on 3 df
-log2(p) of ll-ratio test	569.37




```
# (4)accelerate failure time
```

```
"""
加速失效时间模型分析模块
集成Cox模型的数据处理逻辑，修复列名兼容性和数据验证问题
"""
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifelines import LogLogisticAFTFitter, WeibullAFTFitter, LogNormalAFTFitter

# === 配置 ===
```

```python
DATA_PATH = "data1.csv"
DURATION_COL = 'tenure'
EVENT_COL = 'Churn'
CAT_COLS = ['InternetService', 'Contract']
DIST_TYPE = 'loglogistic'  # 可选: loglogistic, weibull, lognormal
INCLUDE_COLS = ['internetservice_fiber_optic', 'contract_one_year']

# === 工具函数 ===
def clean_col_name(name):
    return name.strip().lower().replace(" ", "_").replace("-", "_")

# === 加载数据 ===
data = pd.read_csv(DATA_PATH)
data[DURATION_COL] = data[DURATION_COL].clip(lower=0.1)
print("原始列:", data.columns.tolist())

# === 验证列 ===
required_cols = [DURATION_COL, EVENT_COL] + CAT_COLS
for col in required_cols:
    if col not in data.columns:
        raise KeyError(f"缺失列: {col}")

# === 检查事件列 ===
if len(data[EVENT_COL].unique()) != 2:
    raise ValueError(f"{EVENT_COL} 必须为二分类变量，目前唯一值: {data[EVENT_COL].unique()}")

# === 编码分类变量 ===
duration_event_df = data[[DURATION_COL, EVENT_COL]].copy()
if CAT_COLS:
    encoded = pd.get_dummies(data[CAT_COLS], prefix_sep='_', drop_first=True)
    encoded.columns = [clean_col_name(col) for col in encoded.columns]
    df_encoded = pd.concat([duration_event_df, encoded], axis=1)
else:
    df_encoded = duration_event_df.copy()

# === 转换事件列为数值型 ===
df_encoded[EVENT_COL] = df_encoded[EVENT_COL].replace({'Yes': 1, 'No': 0}).astype(float)

# === 模型选择 ===
model_map = {
    'loglogistic': LogLogisticAFTFitter,
    'weibull': WeibullAFTFitter,
    'lognormal': LogNormalAFTFitter
}
if DIST_TYPE.lower() not in model_map:
    raise ValueError(f"分布类型 '{DIST_TYPE}' 不支持")
model = model_map[DIST_TYPE.lower()]()

# === 选择特征列 ===
feature_cols = [col for col in df_encoded.columns if col not in [DURATION_COL, EVENT_COL]]
normalized_features = [clean_col_name(col) for col in feature_cols]
selected = []
for col in INCLUDE_COLS:
    col_clean = clean_col_name(col)
    if col_clean in normalized_features:
        matched = feature_cols[normalized_features.index(col_clean)]
```

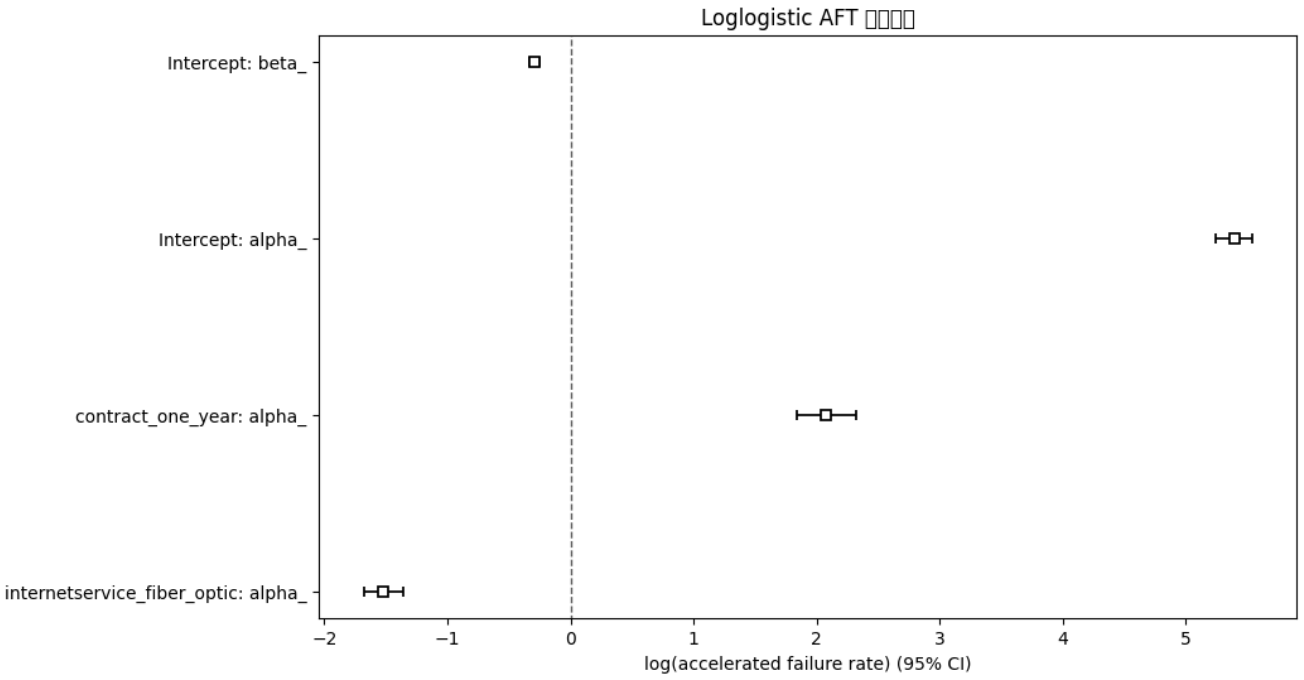
```
        selected.append(matched)
    else:
        print(f"! 忽略无效列: {col}")

print("\n使用特征列:", selected)

# === 拟合模型 ===
model_data = df_encoded[[DURATION_COL, EVENT_COL] + selected]
model.fit(model_data, duration_col=DURATION_COL, event_col=EVENT_COL)

# === 输出结果 ===
print("\n模型摘要:")
print(model.summary)

print(f"\nAIC: {model.AIC_:.2f}")
print(f"一致性指数 (Concordance Index): {model.concordance_index_:.3f}")

# === 可视化参数效应 ===
os.makedirs("aft_report", exist_ok=True)
plt.figure(figsize=(10, 6))
model.plot()
plt.title(f"{DIST_TYPE.capitalize()} AFT 参数效应")
plt.savefig("aft_report/parameter_effects.png")
plt.show()
```

模型摘要:

```
                                coef  exp(coef)  se(coef) \
param  covariate
alpha_ contract_one_year        2.077421   7.983848  0.122979
       internetservice_fiber_optic -1.518619   0.219014  0.082554
       Intercept                5.392630 219.780680  0.075216
beta_  Intercept                -0.291916   0.746831  0.019827
```

```
                                    coef lower 95%  coef upper 95%  \
param  covariate
alpha_ contract_one_year                  1.836387        2.318454
       internetservice_fiber_optic       -1.680421       -1.356817
       Intercept                          5.245210        5.540050
beta_  Intercept                         -0.330777       -0.253055

                                    exp(coef) lower 95%  exp(coef) upper 95%  \
param  covariate
alpha_ contract_one_year                       6.273828            10.159959
       internetservice_fiber_optic             0.186295             0.257479
       Intercept                             189.655686           254.690743
beta_  Intercept                               0.718365             0.776425

                                    cmp to          z             p  \
param  covariate
alpha_ contract_one_year               0.0  16.892523  5.107164e-64
       internetservice_fiber_optic     0.0 -18.395547  1.426151e-75
       Intercept                       0.0  71.695622  0.000000e+00
```

```
beta_  Intercept                       0.0 -14.722879  4.596637e-49

                                     -log2(p)
param  covariate
alpha_ contract_one_year            210.250876
       internetservice_fiber_optic  248.632480
       Intercept                           inf
beta_  Intercept                    160.573898

AIC: 20338.50
一致性指数 (Concordance Index): 0.678
```



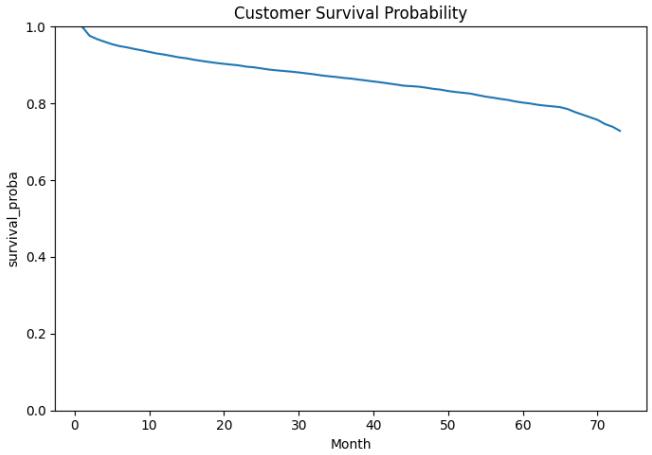

```
# (5)customer lifetime value
```

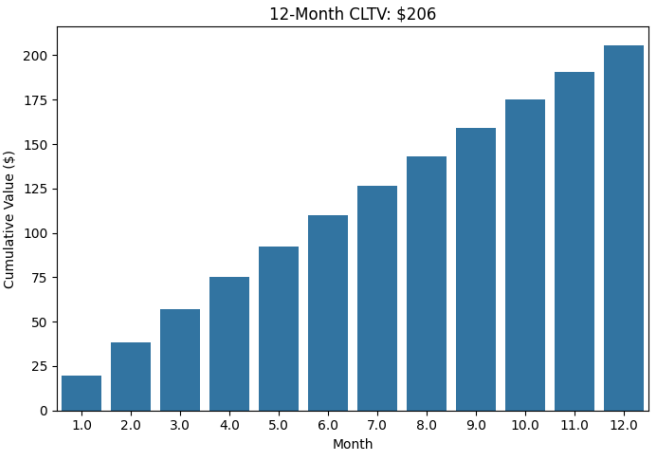
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from lifelines import CoxPHFitter

# === 加载并预处理数据 ===
df = pd.read_csv("data1.csv")

cat_features = ['Dependents', 'InternetService', 'OnlineBackup', 'TechSupport',
'PaperlessBilling']
df_encoded = pd.get_dummies(df, columns=cat_features, prefix=cat_features,
drop_first=True)
df_encoded.columns = [col.replace(' ', '_') for col in df_encoded.columns]

# === 构建 CoxPH 输入数据 ===
cols = [
    'Churn', 'tenure',
```

```
    'Dependents_Yes', 'InternetService_Fiber_optic',
    'OnlineBackup_Yes', 'TechSupport_Yes'
]

missing = [col for col in cols if col not in df_encoded.columns]
if missing:
    print("缺失列:", missing)
    raise KeyError("部分特征编码失败，请检查上方编码步骤")

df_model = df_encoded[cols].copy()
df_model['Churn'] = df_model['Churn'].map({'Yes': 1, 'No': 0}).astype(float)

# === 拟合 Cox 模型 ===
cph = CoxPHFitter()
cph.fit(df_model, duration_col='tenure', event_col='Churn')
print(cph.summary)

# === 示例客户画像（手动设定） ===
sample_customer = {
    'Dependents_Yes': 1,
    'InternetService_Fiber_optic': 0,
    'OnlineBackup_Yes': 0,
    'TechSupport_Yes': 0
}

# === 内联计算 CLTV ===
input_df = pd.DataFrame([{
    'Dependents_Yes': sample_customer.get('Dependents_Yes', 0),
    'InternetService_Fiber_optic': sample_customer.get('InternetService_Fiber_optic', 0),
    'OnlineBackup_Yes': sample_customer.get('OnlineBackup_Yes', 0),
    'TechSupport_Yes': sample_customer.get('TechSupport_Yes', 0)
}])

# 计算生存概率
survival_df = cph.predict_survival_function(input_df)
survival_df.columns = ['survival_proba']
survival_df = survival_df.reset_index().rename(columns={'index': 'month'})
survival_df['month'] += 1  # 从第1个月开始

# 财务参数
monthly_profit = 20
discount_rate = 0.2

# 内联计算 NPV & CLTV
survival_df['expected_profit'] = survival_df['survival_proba'] * monthly_profit
survival_df['npv'] = survival_df['expected_profit'] / ((1 + discount_rate / 12) **
survival_df['month'])
survival_df['cumulative_npv'] = survival_df['npv'].cumsum()

# === 可视化生存与 CLTV ===
plt.figure(figsize=(14, 5))

# 生存概率曲线
plt.subplot(1, 2, 1)
sns.lineplot(data=survival_df, x='month', y='survival_proba')
plt.title('Customer Survival Probability')
```

```
plt.xlabel("Month")
plt.ylim(0, 1)

# 累计 CLTV
plt.subplot(1, 2, 2)
sns.barplot(data=survival_df[:12], x='month', y='cumulative_npv')
plt.title(f"12-Month CLTV: ${survival_df['cumulative_npv'].iloc[11]:.0f}")
plt.xlabel("Month")
plt.ylabel("Cumulative Value ($)")

plt.tight_layout()
plt.show()
```

```
                                 coef  exp(coef)  se(coef)  coef lower 95%  \
covariate
Dependents_Yes               -0.689927   0.501613  0.061918       -0.811283
InternetService_Fiber_optic   1.126852   3.085927  0.051936        1.025060
OnlineBackup_Yes             -0.801981   0.448440  0.054174       -0.908160
TechSupport_Yes              -0.929163   0.394884  0.063619       -1.053854

                             coef upper 95%  exp(coef) lower 95%  \
covariate
Dependents_Yes                     -0.568571             0.444288
InternetService_Fiber_optic         1.228645             2.787262
OnlineBackup_Yes                   -0.695802             0.403265
TechSupport_Yes                    -0.804472             0.348592

                             exp(coef) upper 95%  cmp to          z  \
covariate
Dependents_Yes                          0.566334     0.0 -11.142675
InternetService_Fiber_optic             3.416595     0.0  21.697003
OnlineBackup_Yes                        0.498674     0.0 -14.803833
TechSupport_Yes                         0.447324     0.0 -14.605105

                                      p    -log2(p)
covariate
Dependents_Yes                 7.774836e-29   93.377102
InternetService_Fiber_optic  2.189789e-104  344.349730
OnlineBackup_Yes              1.383673e-49  162.305974
TechSupport_Yes               2.605792e-48  158.070827
```


Customer Survival Probability
survival_proba
Month
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
70


12-Month CLTV: $206
Cumulative Value ($)
Month
200
175
150
125
100
75
50
25
0
1.0
2.0
3.0
4.0
5.0
6.0
7.0
8.0
9.0
10.0
11.0
12.0