Project-1 Report of Spark

Shanli Ouyang

April 2025

Contents

Q1.Security Data Analysis

1.1

Two Spark dataframes(df_http and df_dns)were created. "Q1_data" folder contains the data of 00-05 folders of "http.log.gz" and "dns.log.gz". "ts"column was converted to Timestamp data type, and two temp view named "http_log" and "dns_log"are created.

```python
from pyspark.sql import SparkSession
spark = SparkSession.builder.config('spark.ui.port', 4040).appName("Project_Security").getOrCreate()

http_path = "Q1_data/*/http.log.gz"
dns_path = "Q1_data/*/dns.log.gz"

#读取成为dataframe
df_http = spark.read.json(http_path)
df_dns = spark.read.json(dns_path)

#通过cast转换格式至Timestamp
df_http = df_http.withColumn("ts", df_http["ts"].cast("Timestamp"))

#创建临时视图
df_http.createOrReplaceTempView("http_log")
df_dns.createOrReplaceTempView("dns_log")
```

Figure 1: Code of Question1

1.2

Filter the rows where the status code is 200 and method is GET, sort in a descending order according to the accessed count of the "uri" column With the "http_log" data, using two different method. The two method code and result are shown below.

1.2.1 Spark SQL API

The first one is Spark SQL API.

```python
#Spark SQL API
#过滤、分组和排序
##这里count(*)就是计算每个分组的行数，也就是uri数
##添加了一个筛选无内容uri的做法，防止空uri混入影响数据判断
operated_sql_http = spark.sql("""
    SELECT uri, COUNT(*) AS access_count
    FROM http_log
    WHERE status_code = 200 AND method = 'GET' AND uri != '/'
    GROUP BY uri
    ORDER BY access_count DESC
""")

operated_sql_http.show(5)
```

```
[Stage 40:==============================>                          (1 + 1) / 2]
+--------------------+------------+
|                 uri|access_count|
+--------------------+------------+
|/admin/config.php...|         556|
|  /main.php?logout=1|         194|
|/top.php?stuff=15...|         191|
|            /top.php|         179|
|/main.php?stuff=1...|         172|
+--------------------+------------+
only showing top 5 rows
```

Figure 2: Spark SQL API

1.2.2 Spark Dataframe

The second one is Spark Dataframe.

```
#Spark Dataframe
#过滤、分组和排序
from pyspark.sql.functions import col, count
filtered_http = df_http.filter((col("status_code") == 200) & (col("method") == "GET")& (col("uri") != "/"))
grouped_http = filtered_http.groupBy("uri").agg(count("*").alias("access_count"))
operated_df_http = grouped_http.orderBy(col("access_count").desc())
operated_df_http.show(5)
```

```
[Stage 25:===========================>                          (1 + 1) / 2]
+--------------------+------------+
|                 uri|access_count|
+--------------------+------------+
|/admin/config.php...|         556|
|   /main.php?logout=1|        194|
|/top.php?stuff=15...|         191|
|            /top.php|         179|
|/main.php?stuff=1...|         172|
+--------------------+------------+
only showing top 5 rows
```

Figure 3: Spark Dataframe

1.3

Using Spark SQL to join the "http_log" and calculate the percentage of "proto"="tcp" for each "dns_log" tables by "uid" column.

```
##join
joined_df = spark.sql("""
    Select *
    From http_log
    Join dns_log
    On http_log.uid = dns_log.uid
""")
joined_df.show(5)
```

```
+----+---------+---------+---------+---------+------+--------------+-----------+---------------+------+-------+--------+-----------------+--------------+---
--------+---------------+-----------------+-----------+----------+----+-----------+---+---+---+----------+--------+-------+---+---+---+---+----+---+-----
--+---------+---------+---------+---------+-----+------+-----------+-----+----------+-----+-----+----------+--------+---+--------+---+---+
|host|id.orig_h|id.orig_p|id.resp_h|id.resp_p|method|orig_filenames|orig_fuids|orig_mime_types|origin|proxied|referrer|request_body_len|resp_filenames|re
sp_fuids|resp_mime_types|response_body_len|status_code|status_msg|tags|trans_depth| ts|uid|uri|user_agent|username|version| AA| RA| RD| TC|TTLs|  Z|answe
rs|id.orig_h|id.orig_p|id.resp_h|id.resp_p|proto|qclass|qclass_name|qtype|qtype_name|query|rcode|rcode_name|rejected|rtt|trans_id| ts|uid|
+----+---------+---------+---------+---------+------+--------------+-----------+---------------+------+-------+--------+-----------------+--------------+---
--------+---------------+-----------------+-----------+----------+----+-----------+---+---+---+----------+--------+-------+---+---+---+---+----+---+-----
--+---------+---------+---------+---------+-----+------+-----------+-----+----------+-----+-----+----------+--------+---+--------+---+---+
+----+---------+---------+---------+---------+------+--------------+-----------+---------------+------+-------+--------+-----------------+--------------+---
--------+---------------+-----------------+-----------+----------+----+-----------+---+---+---+----------+--------+-------+---+---+---+---+----+---+-----
--+---------+---------+---------+---------+-----+------+-----------+-----+----------+-----+-----+----------+--------+---+--------+---+---+
```

Figure 4: Join code and result

But the result is, we have nothing in the joint dataframe. It is unable to perform the operation of calculating the percentage of "proto"="tcp" for each group grouped by "uid" in the "dns_log" tables. The code is below.

Q2.Survival Analysis

2.1

I have re-write all the pyspark code, and those code are provided in my code submit. The following figure 6 showed what I wrote.

2.2

In this quesion, I will explain the use of those five survival analysis process, and the result will be as well provided.

```
##calculate
calculate_http = spark.sql("""
    SELECT uri, ROUND((COUNT(proto = 'tcp')/COUNT(*))*100,2) AS tcp_percentage
    FROM http_log
    WHERE status_code = 200 AND method = 'GET' AND uri != '/'
    GROUP BY uri
    ORDER BY access_count DESC
""")
calculate_http.show(5)
##http_log的数据中不包含proto列，但http_log和dns_log合成后的结果为空，故无法实验
```

Figure 5: Calculate code

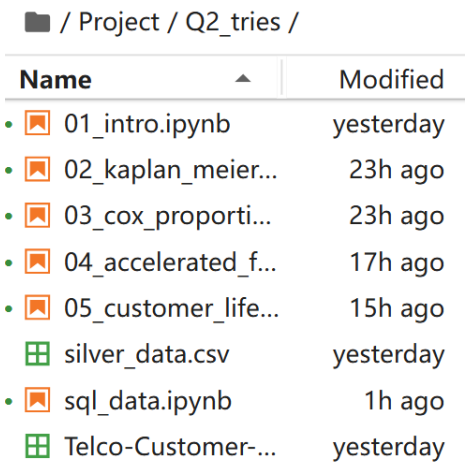


Figure 6: Codes

2.2.1 Intro

In "01_intro.ipynb", through relevant operations on the columns of raw data, the "silver_data.csv" has been processed for subsequent functionality, as shown in Figure 7.

	customerID	gender	seniorCitizen	partner	dependents
1	7590-VHVEG	Female	0.0	Yes	No
2	3668-QPYBK	Male	0.0	No	No
3	9237-HQITU	Female	0.0	No	No
4	9305-CDSKC	Female	0.0	No	No
5	1452-KIOVK	Male	0.0	No	Yes
6	6713-OKOMC	Female	0.0	No	No
7	7892-POOKP	Female	0.0	Yes	No
8	9763-GRSKD	Male	0.0	Yes	Yes
9	0280-XJGEX	Male	0.0	No	No
10	5129-JLPIS	Male	0.0	No	No
11	4190-MFLUW	Female	0.0	Yes	Yes
12	4183-MYFRB	Female	0.0	No	No
13	8779-QRDMV	Male	1.0	No	No
14	6322-HRPFA	Male	0.0	Yes	Yes
15	6865-JZNKO	Female	0.0	No	No
16	6467-CHFZW	Male	0.0	Yes	Yes
17	8665-UTDHZ	Male	0.0	Yes	Yes
18	8773-HHUOZ	Female	0.0	No	Yes
19	4929-XIHVW	Male	1.0	Yes	No

Figure 7: silver data

2.2.2 Kaplan Meier

In "02_kaplan_meier.ipynb", the procedure mainly conducts a survival analysis of "telecom customer churn" data, revealing the impact of various features (e.g., gender, internet service type, senior citizen status) on customer churn. Through visualized survival curves and statistical tests, it provides an in-depth understanding of customer churn behavior.

After short data loading and preprocessing, a Kaplan-Meier model (KaplanMeierFitter) is initialized and fitted using the "tenure" and "churn" data. An overall survival curve(Figure 8) is plotted to show the probability of survival at different time points for the entire customer population.

Then,A function "plot_km(col)" is defined to group customers by a specified column (e.g., gender, internetService) and plot the survival curves for each group. Also, a function "print_logrank(col)" is defined to perform a Log-rank test for a specified column, determining whether there are significant differences in survival curves between groups.

The two function can be used to analysis any of the columns, so I select two columns to give some examples: as shown in Figure 9,the survival curve of different groups are clearly shown on the Survival Prob over Timeline graphs. And the significance tests show that the p-value of gender group test is larger than 0.05, which means the survival curve difference between Female and Male

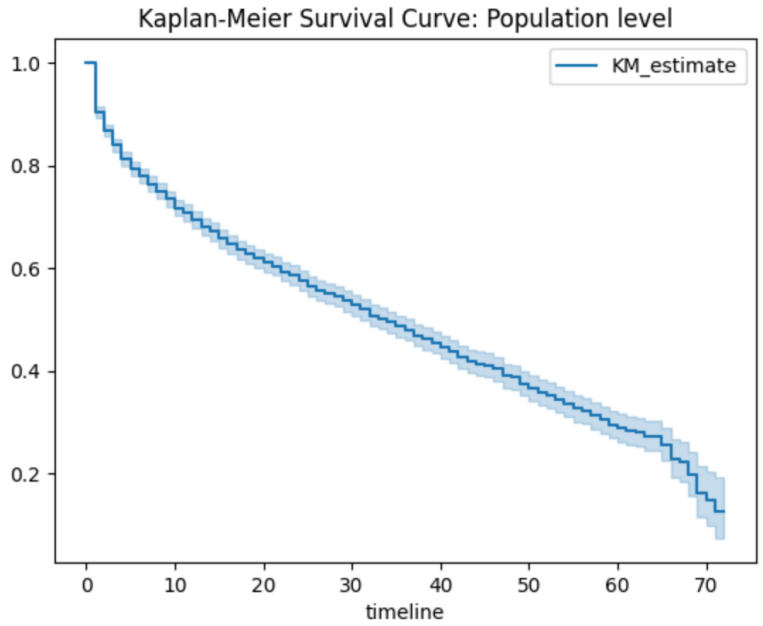


Figure 8: Survival Curve of KaplanMeierFitter

is not significant; instead, in the partner group test, the p-value is very small, indicating a significant difference between having or not having a partner.

Those analysis of the two functions helps observe the differences in survival probabilities among different customer groups based on specific features, providing insights into the significance of features on customer churn.

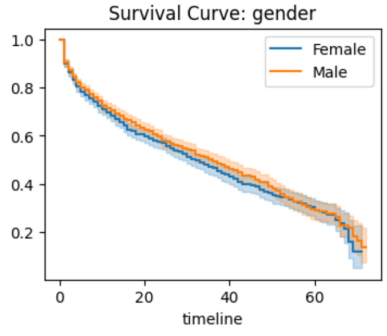
At last, a function "get_survival_probs(col, val)" is defined to calculate the survival probabilities for a specific group (e.g., internetService=DSL) at specified time points.The results are returned as a DataFrame, showing the survival probabilities for the group from time 0 to 9.

2.2.3 Cox Proportional Hazards

After data preprocessing, a Cox Proportional Hazards model (CoxPHFitter) is initialized, using tenure (customer tenure) as the time variable and churn (whether the customer churned) as the event variable to fit the model. The model summary is including coefficients, hazard ratios, and p-values for each feature, as shown in Figure 10.

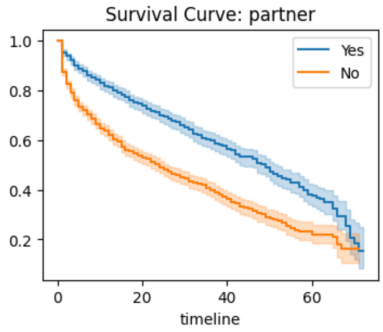
A hazard ratio plot is generated to visually display the impact of each feature on customer churn, as shown in Figure 11.

The proportional hazards assumption is tested using the "check_assumptions method", and relevant plots are generated to validate the assumption. The assumption result is shown in Figure 12, the p-value indicates whether the



		test_statistic	p	-log2(p)
Female	**Male**	1.61011	0.204476	2.289995

(a) Over Gender Groups



		test_statistic	p	-log2(p)
No	**Yes**	257.844159	5.063437e-58	190.331712

(b) Over Partners Groups

Figure 9: Survival Curves and Significance Test

model	lifelines.CoxPHFitter
duration col	'tenure'
event col	'churn'
baseline estimation	breslow
number of observations	3351
number of events observed	1556
partial log-likelihood	-11178.40
time fit was run	2025-04-11 09:59:50 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
dependents_Yes	-0.10	0.90	0.07	-0.24	0.03	0.79	1.03	0.00	-1.52	0.13	2.95
internetService_DSL	-0.04	0.96	0.06	-0.15	0.07	0.86	1.07	0.00	-0.78	0.44	1.19
onlineBackup_Yes	-0.35	0.71	0.06	-0.46	-0.23	0.63	0.79	0.00	-6.06	<0.005	29.49
techSupport_Yes	-0.21	0.81	0.07	-0.34	-0.08	0.71	0.92	0.00	-3.11	<0.005	9.06

Concordance	0.64
Partial AIC	22364.80
log-likelihood ratio test	57.57 on 4 df
-log2(p) of ll-ratio test	36.63

Figure 10: CPH Model Summary

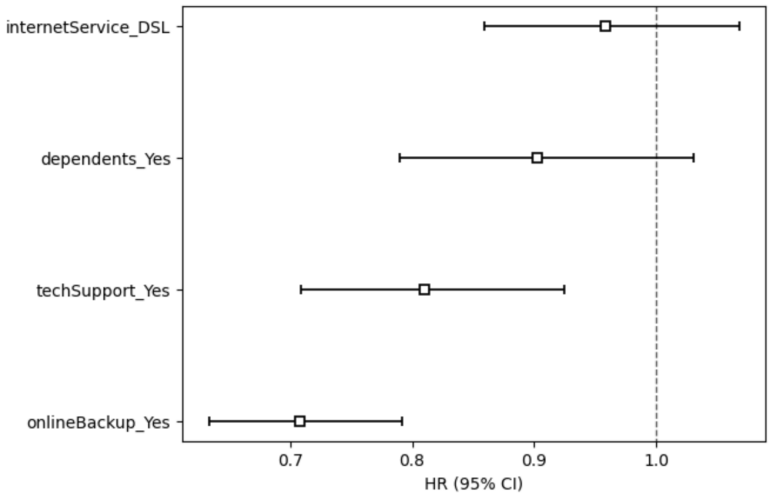


Figure 11: HR Plot of CPH Model

proportional hazards assumption holds.

Also, the result of the assumption can be intuitively seen in the plot. Two examples of two variables(group name) are provided in Figure 13.

In the "dependent_Yes" plot, p_value over rank is larger than 0.05, meaning we cannot reject the null hypothesis: the survival curve of dependent case or not are proportional, meaning that the difference is not significant. On the contrary, we can see that in "onlineBackup_Yes" plot, the p_values are all smaller than 0.05, so the difference of survival curve between groups are significant.

After that, a Kaplan-Meier model (KaplanMeierFitter) is initialized and fitted using tenure and churn data. A function "plot_km_loglog(col)" is defined to plot the log-log survival curves for specified columns (e.g., onlineBackup, dependents) to group customers by specific features. The log-log plots are used to observe whether the survival curves of different feature groups are parallel. If the curves are parallel, the proportional hazards assumption holds; if not, the assumption is violated. Some examples are shown in Figure 14.

2.2.4 Accelerated Failure Time

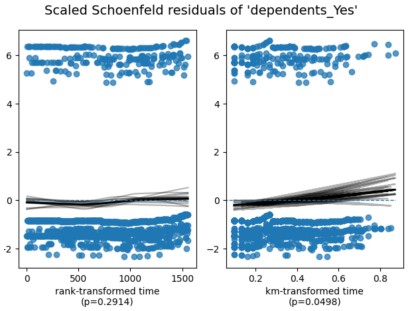
After data preprocessing, a LogLogisticAFTFitter model is initialized to analyze time-dependent risk changes in customer "churn". The model is fitted using tenure (customer tenure) as the time variable and churn (whether the customer churned) as the event variable. The median survival time and parameter summary are printed to evaluate the impact of different features on customer churn risk, as shown in Figure 14.

A log plot to show the 95% CI of different variables are shown in Figure 14.

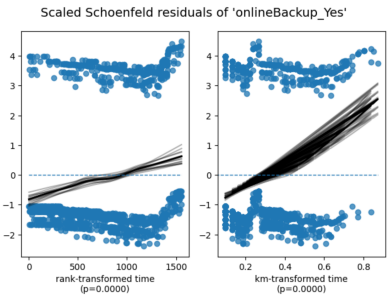
null_distribution	chi squared
degrees_of_freedom	1
model	<lifelines.CoxPHFitter: fitted with 3351 total...
test_name	proportional_hazard_test

		test_statistic	p	-log2(p)
dependents_Yes	**km**	3.85	0.05	4.33
	rank	1.11	0.29	1.78
internetService_DSL	**km**	48.21	<0.005	37.93
	rank	15.59	<0.005	13.63
onlineBackup_Yes	**km**	106.63	<0.005	80.63
	rank	47.80	<0.005	37.62
techSupport_Yes	**km**	13.68	<0.005	12.17
	rank	10.23	<0.005	9.50

Figure 12: Proportional Hazards Assumption Summary

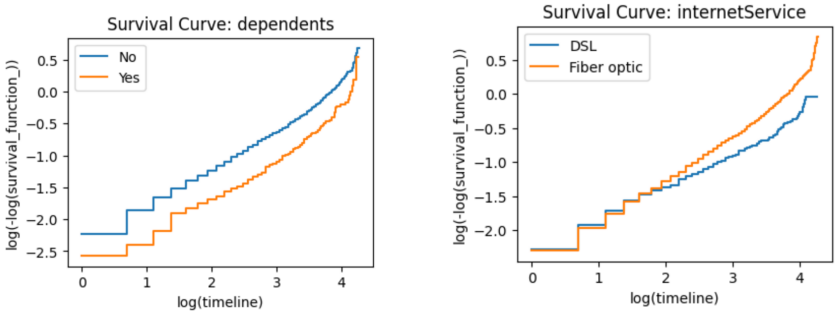


(a) Over Dependent_Yes



(b) Over OnlineBackup_Yes

Figure 13: Proportional Hazards Assumption Result



(a) Over dependent (b) Over internetService

Figure 14: Survival Curve of log(timeline)

Median Survival Time:135.51

model	lifelines.LogLogisticAFTFitter
duration col	'tenure'
event col	'churn'
number of observations	3351
number of events observed	1556
log-likelihood	-6838.36
time fit was run	2025-04-11 15:58:21 UTC

		coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
alpha_	deviceProtection_Yes	0.48	1.62	0.07	0.35	0.62	1.41	1.86	0.00	6.88	<0.005	37.25
	internetService_DSL	0.38	1.47	0.08	0.23	0.53	1.26	1.71	0.00	4.98	<0.005	20.59
	multipleLines_Yes	0.66	1.94	0.07	0.53	0.80	1.70	2.22	0.00	9.64	<0.005	70.70
	onlineBackup_Yes	0.81	2.25	0.07	0.68	0.95	1.97	2.59	0.00	11.63	<0.005	101.50
	onlineSecurity_Yes	0.86	2.37	0.09	0.69	1.03	2.00	2.80	0.00	10.12	<0.005	77.60
	partner_Yes	0.68	1.97	0.07	0.55	0.81	1.73	2.24	0.00	10.21	<0.005	78.93
	paymentMethod_Bank transfer (automatic)	0.74	2.10	0.09	0.56	0.92	1.75	2.51	0.00	8.05	<0.005	50.07
	paymentMethod_Credit card (automatic)	0.80	2.22	0.10	0.61	0.99	1.84	2.68	0.00	8.36	<0.005	53.81
	techSupport_Yes	0.69	1.99	0.09	0.52	0.86	1.68	2.36	0.00	7.90	<0.005	48.37
	Intercept	1.59	4.91	0.07	1.46	1.72	4.32	5.58	0.00	24.47	<0.005	436.88
beta_	Intercept	0.12	1.13	0.02	0.08	0.16	1.08	1.17	0.00	5.71	<0.005	26.42

Concordance	0.73
AIC	13698.72
log-likelihood ratio test	877.49 on 9 df
-log2(p) of ll-ratio test	605.78

Figure 15: LogLogisticAFTFitter Model Summary

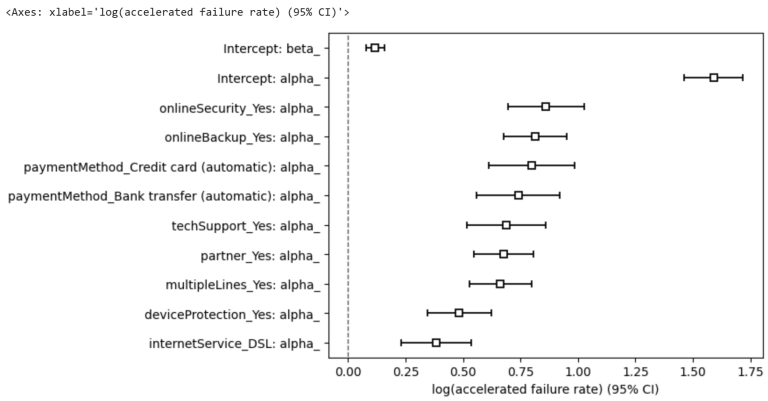
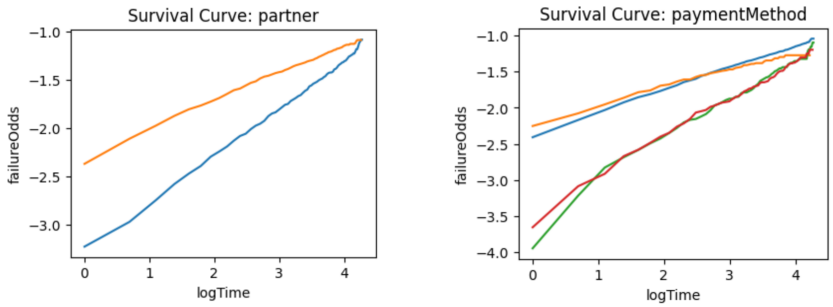


Figure 16: LogLogisticAFTFitter log(accelerated failure rate) (95% CI)

After that, as the previous methods, a Kaplan-Meier model (KaplanMeierFitter) is initialized and fitted using tenure and churn data. A function is defined to plot the log-odds survival curves for specified columns (e.g., partner, multipleLines). The log-odds survival curve is generated by mathematically transforming the survival function, showing the change in customer churn risk over time for different feature groups. Two examples are provided in Figure 17.



(a) Over partner

(b) Over paymentMethod

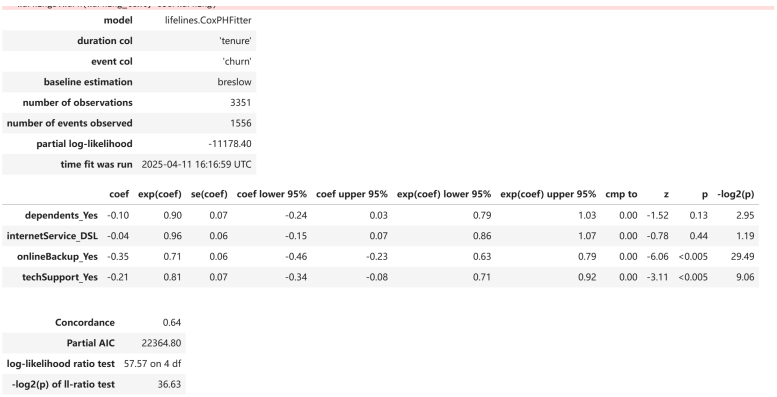
Figure 17: Survival Curve of Timeline

The log-odds survival curves visually demonstrate whether there are significant differences in customer churn risk across different feature groups and how these risks change over time.

2.2.5 Customer Life Time Value

In this procedure, a Cox Proportional Hazards model (CoxPHFitter) is initialized, using tenure (customer tenure) as the time variable and churn (whether the customer churned) as the event variable to fit the model. The model summary

is printed, including coefficients, hazard ratios, and p-values for each feature, to evaluate the impact of features on customer churn. The summary is shown in Figure 18.



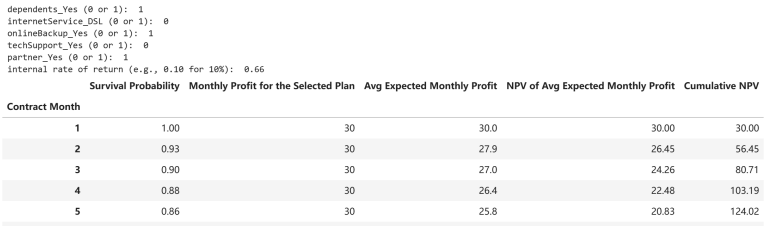
model	lifelines.CoxPHFitter
duration col	'tenure'
event col	'churn'
baseline estimation	breslow
number of observations	3351
number of events observed	1556
partial log-likelihood	-11178.40
time fit was run	2025-04-11 16:16:59 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	cmp to	z	p	-log2(p)
dependents_Yes	-0.10	0.90	0.07	-0.24	0.03	0.79	1.03	0.00	-1.52	0.13	2.95
internetService_DSL	-0.04	0.96	0.06	-0.15	0.07	0.86	1.07	0.00	-0.78	0.44	1.19
onlineBackup_Yes	-0.35	0.71	0.06	-0.46	-0.23	0.63	0.79	0.00	-6.06	<0.005	29.49
techSupport_Yes	-0.21	0.81	0.07	-0.34	-0.08	0.71	0.92	0.00	-3.11	<0.005	9.06

Concordance	0.64
Partial AIC	22364.80
log-likelihood ratio test	57.57 on 4 df
-log2(p) of ll-ratio test	36.63

Figure 18: Cox Proportional Hazards model Summary

After that, a function "get_user_input()" is defined to collect user input for customer features (e.g., whether the customer has dependents, uses DSL, etc.). Then, a function "get_payback_df()" is defined to calculate the cumulative NPV and survival probability for different contract durations. The cumulative NPV is computed by discounting the expected monthly profit using the IRR and summing it over time. The survival probability is predicted by the Cox model and associated with the contract duration. And one of the result of prediction is shown in the Table 19.

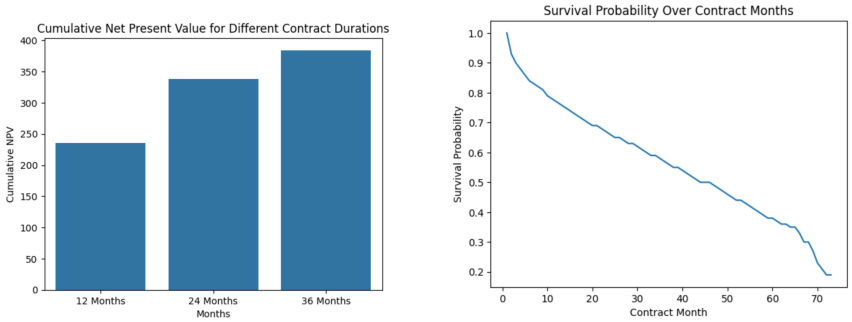


```
dependents_Yes (0 or 1):  1
internetService_DSL (0 or 1):  0
onlineBackup_Yes (0 or 1):  1
techSupport_Yes (0 or 1):  0
partner_Yes (0 or 1):  1
internal rate of return (e.g., 0.10 for 10%):  0.66
```

Contract Month	Survival Probability	Monthly Profit for the Selected Plan	Avg Expected Monthly Profit	NPV of Avg Expected Monthly Profit	Cumulative NPV
1	1.00	30	30.0	30.00	30.00
2	0.93	30	27.9	26.45	56.45
3	0.90	30	27.0	24.26	80.71
4	0.88	30	26.4	22.48	103.19
5	0.86	30	25.8	20.83	124.02

Figure 19: Prediction example

Then, Seaborn and Matplotlib are used to visualize the cumulative NPV and survival probability over contract duration. A bar chart shows the cumulative NPV for different contract durations (e.g., 12 months, 24 months, 36 months), and a line chart shows the survival probability over contract months, which are shown in the Figure 20.



(a) cumulative NPV (b) Survival Probability

Figure 20: Visualization over contract duration

2.2.6 Conclusion of all codes

Above all, we can conclude that: If you are concerned with the time until an event occurs: Use the Accelerated Failure Time (AFT) Model.The AFT model directly models the time to the event and is suitable for analyzing how covariates influence the time until an event (e.g., customer churn).

If you are concerned with the impact of features on the risk of an event: Use the Cox Proportional Hazards (CPH) Model. The CPH model estimates the hazard ratio for each covariate, providing insights into how features affect the risk of an event occurring over time.

If you are only concerned with the distribution of survival probabilities: Use the Kaplan-Meier Estimator (KMF). The Kaplan-Meier method estimates the survival function over time, making it ideal for visualizing and analyzing the probability of survival without considering covariates.

2.3

In this question, I was tasked with using Large Language Models (LLMs) to generate a set of statements related to database data processing. Subsequently, I was required to identify potential errors within these generated statements and analyze the possible reasons for these errors.

First, I upload a dataframe "silver" to my database "spark_project". The schema is shown in Figure 21 below using SQL code.

I did all my operation on the termial, but in order to facilitate the submission of code and presentation, I created a temporary script to write the SQL code.

2.3.1

The first wrong code that the LLMs gave me was shown in Figure **??**, meanwhile my question was: "I want to query customers with tenure column content between 24 and 60 months, and select the content display for the customerID and tenure columns." The right one was in Figure 22b. It is clear that the mistake

```
mysql> use spark_project;
Database changed
mysql> DESCRIBE spark_project.silver;
ERROR 4031 (HY000): The client was disconnected by the server because of inactiv
No connection. Trying to reconnect...
Connection id:    72
Current database: spark_project

+-------------------+----------+------+-----+---------+-------+
| Field             | Type     | Null | Key | Default | Extra |
+-------------------+----------+------+-----+---------+-------+
| customerID        | longtext | YES  |     | NULL    |       |
| gender            | longtext | YES  |     | NULL    |       |
| seniorCitizen     | double   | YES  |     | NULL    |       |
| partner           | longtext | YES  |     | NULL    |       |
| dependents        | longtext | YES  |     | NULL    |       |
| tenure            | double   | YES  |     | NULL    |       |
| phoneService      | longtext | YES  |     | NULL    |       |
| multipleLines     | longtext | YES  |     | NULL    |       |
| internetService   | longtext | YES  |     | NULL    |       |
| onlineSecurity    | longtext | YES  |     | NULL    |       |
| onlineBackup      | longtext | YES  |     | NULL    |       |
| deviceProtection  | longtext | YES  |     | NULL    |       |
| techSupport       | longtext | YES  |     | NULL    |       |
| streamingTV       | longtext | YES  |     | NULL    |       |
| streamingMovies   | longtext | YES  |     | NULL    |       |
| contract          | longtext | YES  |     | NULL    |       |
| paperlessBilling  | longtext | YES  |     | NULL    |       |
| paymentMethod     | longtext | YES  |     | NULL    |       |
| monthlyCharges    | double   | YES  |     | NULL    |       |
| totalCharges      | double   | YES  |     | NULL    |       |
| churn             | int      | YES  |     | NULL    |       |
+-------------------+----------+------+-----+---------+-------+
21 rows in set (0.01 sec)
```

Figure 21: Schema of Silver Table in Database

```
##通过临时视图简略地复现我所完成的LLM代码
df.createOrReplaceTempView("silver")
select_months = spark.sql("""
SELECT customerID, tenure
FROM silver
WHERE tenure > 24 AND tenure < 60;
""")
select_months.show(5)
##这里LLM没有理解我的指令，因此没有取闭区间
```

```
+----------+------+
|customerID|tenure|
+----------+------+
|7892-POOKP|  28.0|
|0280-XJGEX|  49.0|
|5129-JLPIS|  25.0|
|6322-HRPFA|  49.0|
|6865-JZNKO|  30.0|
+----------+------+
only showing top 5 rows
```

(a) first wrong code

```
##正确情况
select_months_true = spark.sql("""
SELECT customerID, tenure
FROM silver
WHERE tenure BETWEEN 24 AND 60;
""")
select_months_true.show(5)
```

```
+----------+------+
|customerID|tenure|
+----------+------+
|7892-POOKP|  28.0|
|0280-XJGEX|  49.0|
|5129-JLPIS|  25.0|
|6322-HRPFA|  49.0|
|6865-JZNKO|  30.0|
+----------+------+
only showing top 5 rows
```

(b) first right code

Figure 22: First wrong try

reason was the LLM didn't got my idea. When I say 24 to 60, I mean both 24 and 60 are included, by the code LLMs gave me didn't.

2.3.2

The second wrong code that the LLMs gave me was shown in Figure 23a, meanwhile my question was in Figure 24, in Chinese. What we met was after putting the SQL code provided to the terminal, the terminal throwed an error of "You cannot nest a window function in the specification of window '¡unnsamed window¿'."

(a) second wrong code

(b) first right code

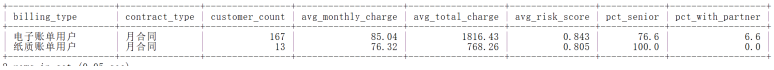
Figure 23: Second wrong try

After splitting the risk score calculation into a separate Common Table Expression (CTE) named "risk_scoring", calculating the percentile rankings in a distinct CTE called "risk_with_rank", and finally replacing window functions with subqueries when computing the minimum and maximum monthly charge values, we got the right code in Figure 23b. and the answer is in Figure 25.

2.4

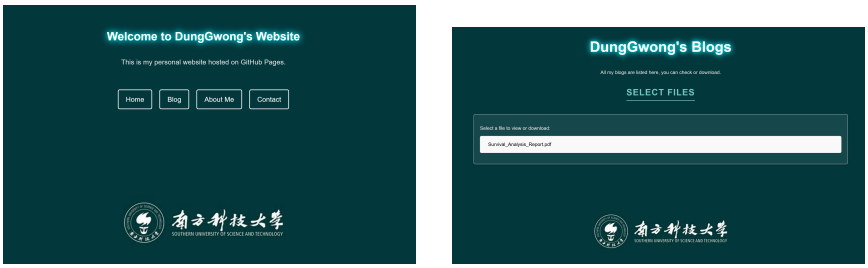
In this part, I create my personal website using github pages and deployed my custom domain on this site. I have uploaded my "Survival_Analysis_Report.pdf". The website url is "https://dunggwong.github.io/",and the web interface in shown in Figure 26.

Figure 24: The second question



billing_type	contract_type	customer_count	avg_monthly_charge	avg_total_charge	avg_risk_score	pct_senior	pct_with_partner
电子账单用户	月合同	167	85.04	1816.43	0.843	76.6	6.6
纸质账单用户	月合同	13	76.32	768.26	0.805	100.0	0.0

Figure 25: The right result



(a) index page (b) blog page

Figure 26: My own blog website