```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from lifelines import WeibullAFTFitter, LogNormalAFTFitter, LogLogisticAFTFitter, ExponentialFitter
# from Exponential import ExponentialAFTFitter
from lifelines.utils import k_fold_cross_validation
import seaborn as sns
import warnings
```

```
warnings.filterwarnings("ignore")
```

```
data_path = 'telco.csv'
raw_data = pd.read_csv(data_path)
```

```
def process_data(data):
    data = data.copy()
    data.drop(['ID'], axis=1, inplace=True)
    cols = ['region', 'retire', 'marital', 'ed', 'gender', 'voice', 'internet', 'custcat', 'churn', 'forward']
    data = data.copy()
    data = pd.get_dummies(data, columns=cols, drop_first=True)
    data = data.rename(columns={'churn_Yes': 'churn'})
    return data
```

```
data = process_data(raw_data)
data
```

:s	marital_Unmarried	ed_Did not complete high school	ed_High school degree	ed_Post-undergraduate degree	ed_Some college	gender_Male	voice_Yes
0	0	0	0	0	0	1	0
0	0	0	0	1	0	1	1
0	0	1	0	0	0	0	0
0	1	0	1	0	0	0	0
0	0	1	0	0	0	1	0
...	...	...	...	...	...	...	...
0	1	0	0	0	1	0	0
0	1	0	0	1	0	0	0
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	0	0	0	0	1	0	0

```
weibull_model = WeibullAFTFitter()
log_norm_model = LogNormalAFTFitter()
log_logistic_model = LogLogisticAFTFitter()
exponential_model = ExponentialAFTFitter()
```

```
weibull = weibull_model.fit(data, duration_col='tenure', event_col='churn')
weibull_prediction = weibull.predict_survival_function(data).T
weibull_prediction_avg = weibull_prediction.mean()
weibull.print_summary()
```

model	lifelines.WeibullAFTFitter
duration col	'tenure'
event col	'churn'
number of observations	1000
number of events observed	274
log-likelihood	-1462.17
time fit was run	2023-11-25 15:54:07 UTC

		coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%
lambda_	**address**	0.04	1.04	0.01	0.02	0.06	1.02	1.06
	age	0.03	1.03	0.01	0.01	0.04	1.01	1.04
	custcat_E-service	0.98	2.66	0.16	0.67	1.28	1.96	3.61
	custcat_Plus service	0.74	2.10	0.19	0.36	1.12	1.44	3.06
	custcat_Total service	1.00	2.71	0.21	0.58	1.41	1.78	4.11
	ed_Did not complete high school	0.44	1.55	0.19	0.06	0.82	1.06	2.27
	ed_High school degree	0.32	1.38	0.15	0.03	0.61	1.03	1.83
	ed_Post-undergraduate degree	0.22	1.25	0.19	-0.15	0.60	0.86	1.82
	ed_Some college	0.25	1.29	0.14	-0.03	0.54	0.97	1.71
	forward_Yes	-0.10	0.91	0.15	-0.39	0.19	0.68	1.21
	gender_Male	0.00	1.00	0.10	-0.20	0.21	0.82	1.23
	income	0.00	1.00	0.00	-0.00	0.00	1.00	1.00
	internet_Yes	-0.77	0.46	0.14	-1.04	-0.50	0.35	0.61
	marital_Unmarried	-0.35	0.71	0.10	-0.55	-0.14	0.58	0.87
	region_Zone 2	-0.06	0.94	0.13	-0.31	0.19	0.73	1.21
	region_Zone 3	0.12	1.12	0.13	-0.13	0.36	0.87	1.44

```
log_norm = log_norm_model.fit(data, duration_col='tenure', event_col='churn')
log_norm_prediction = log_norm.predict_survival_function(data).T
log_norm_prediction_avg = log_norm_prediction.mean()
log_norm.print_summary()
```

model	lifelines.LogNormalAFTFitter
duration col	'tenure'
event col	'churn'
number of observations	1000
number of events observed	274
log-likelihood	-1457.01
time fit was run	2023-11-25 15:54:10 UTC

		coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%
mu_	**address**	0.04	1.04	0.01	0.03	0.06	1.03	1.06
	age	0.03	1.03	0.01	0.02	0.05	1.02	1.05
	custcat_E-service	1.07	2.90	0.17	0.73	1.40	2.08	4.06
	custcat_Plus service	0.92	2.52	0.22	0.50	1.35	1.65	3.85
	custcat_Total service	1.20	3.32	0.25	0.71	1.69	2.03	5.42
	ed_Did not complete high school	0.37	1.45	0.20	-0.02	0.77	0.98	2.16
	ed_High school degree	0.32	1.37	0.16	-0.00	0.64	1.00	1.89
	ed_Post-							

```
log_logistic = log_logistic_model.fit(data, duration_col='tenure', event_col='churn')
log_logistic_prediction = log_logistic.predict_survival_function(data).T
log_logistic_prediction_avg = log_logistic_prediction.mean()
log_logistic.print_summary()
```

model	lifelines.LogLogisticAFTFitter
duration col	'tenure'
event col	'churn'
number of observations	1000
number of events observed	274
log-likelihood	-1458.10
time fit was run	2023-11-25 15:54:11 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_	**address**	0.04	1.04	0.01	0.02	0.06	1.02	1.06	0.00	4.42	<0.005	16.60

Exponential

```
# exponential = exponential_model.fit(data, duration_col='tenure', event_col='churn')
# exponential_prediction = exponential.predict_survival_function(data).T
# exponential_prediction_avg = exponential_prediction.mean()
# exponential.print_summary()
```

Comparing With AIC

_ degree -0.02 0.98 0.22 -0.45 0.40 0.64 1.49 0.00 -0.11 0.91 0.13

```
# print(f'Exponential AIC: {exponential.AIC_}')
print(f'Log-Normal AIC: {log_norm.AIC_}')
print(f'Log-Logistic AIC: {log_logistic.AIC_}')
print(f'Weibull AIC: {weibull.AIC_}')
#'Exponential': exponential.AIC_,
scores = {'Log-normal': log_norm.AIC_, 'Log-logistic': log_logistic.AIC_, 'Weibull': weibull.AIC_}
print(f'\nThe best model based on AIC scores is: \033[1m{min(scores, key=scores.get)}\033[0m')
```

```
Log-Normal AIC: 2954.0240102517128
Log-Logistic AIC: 2956.2085614433336
Weibull AIC: 2964.3432480838806

The best model based on AIC scores is: Log-normal
```

Other than the AIC score and the plots, there are some other important factors to consider before choosing the best model.

When we look at the number of parameters we can understand the complexity of the model. For example, the exponential model has only 1 parameter, while weibull has 2, and Log Normal and Log Logistic models both have 3.

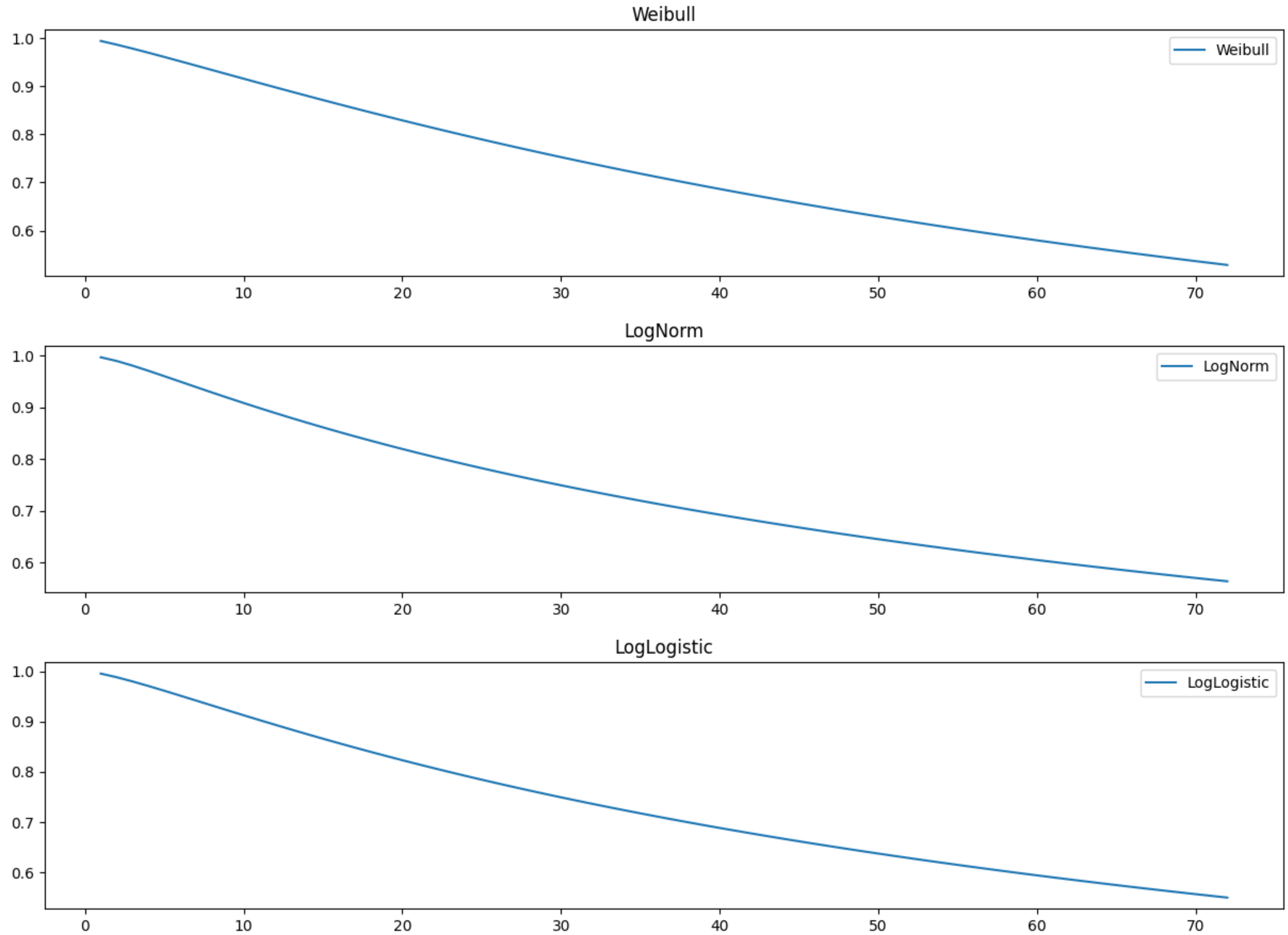
Another criteria is the Hazard Rate. In this case Weibull model is preffered over the other ones as it has the ability to capture both increasing and decreasing hazard rates.

I am going to trust the AIC score and go with the best performing model which is the Log-Normal model.

```
plt.figure(figsize=(15, 3))
plt.plot(weibull_prediction_avg, label='Weibull')
plt.legend()
plt.title('Weibull')
plt.show()

plt.figure(figsize=(15, 3))
plt.plot(log_norm_prediction_avg, label='LogNorm')
plt.legend()
plt.title('LogNorm')
plt.show()

plt.figure(figsize=(15, 3))
plt.plot(log_logistic_prediction_avg, label='LogLogistic')
plt.legend()
plt.title('LogLogistic')
plt.show()
```




```
plt.figure(figsize=(15,9))
plt.plot(weibull_prediction_avg, label='Weibull')
plt.plot(log_norm_prediction_avg, label='LogNorm')
plt.plot(log_logistic_prediction_avg, label='LogLogistic')
# plt.plot(exponential_prediction_avg, label='Exponential')
plt.legend()
plt.show()
```

We will keep the features that are important for calculating the CLV

```
significant_columns = ["address", "age", "internet_Yes", "marital_Unmarried", "tenure", "churn", "custcat_E-service", "custcat_Plus service"
```

```
dropped_data = data[significant_columns]
dropped_data
```

	address	age	internet_Yes	marital_Unmarried	tenure	churn	custcat_E-service	custcat_Plus service	custcat_Total service	voice_Yes
0	9	44	0	0	13	1	0	0	0	0
1	7	33	0	0	11	1	0	0	1	1
2	24	52	0	0	68	0	0	1	0	0
3	12	33	0	1	33	1	0	0	0	0
4	9	30	0	0	23	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...	...
995	0	39	0	1	10	0	0	0	0	0
996	2	34	0	1	7	0	0	0	0	0
997	40	59	1	1	67	0	0	0	1	1
998	18	49	0	1	70	0	0	1	0	1
999	7	36	1	0	50	1	1	0	0	0

1000 rows × 10 columns

```
log_norm = log_norm_model.fit(dropped_data, duration_col='tenure', event_col='churn')
log_norm_prediction = log_norm.predict_survival_function(dropped_data).T
log_norm_prediction_avg = log_norm_prediction.mean()
log_norm.print_summary()
```

model	lifelines.LogNormalAFTFitter
duration col	'tenure'
event col	'churn'
number of observations	1000
number of events observed	274
log-likelihood	-1462.10
time fit was run	2023-11-25 16:01:39 UTC

					coef	coef	exp(coef)	exp(coef)	cmp			-
mu	**address**	0.04	1.04	0.01	0.03	0.06	1.03	1.06	0.00	4.84	<0.005	19.56
	custcat_Plus	0.82	2.28	0.17	0.49	1.15	1.63	3.17	0.00	4.85	<0.005	19.66
	custcat_Total	1.01	2.75	0.21	0.60	1.42	1.83	4.15	0.00	4.83	<0.005	19.52
	Intercept	2.53	12.62	0.24	2.06	3.01	7.84	20.30	0.00	10.45	<0.005	82.47

CLV

_

```
clv_data = log_norm_prediction.copy()
```

Taking some conventional numbers for margin and r

```
margin = 1000
sequence = range(1,len(clv_data.columns)+1)
r = 0.1
```

log-likelihood ratio test 280.83 on 8 df

```
for i in sequence:
    clv_data.loc[:, i] = clv_data.loc[:, i]/((1+r/12)**(sequence[i-1]-1))
```

```
clv_data["CLV"] = margin * clv_data.sum(axis = 1)
clv_data
```

0	5.0	6.0	7.0	8.0	9.0	10.0	...	64.0	65.0	66.0	67.0	68.0	69.0	70.0	71.
3	0.954194	0.940967	0.927370	0.913526	0.899533	0.885469	...	0.363526	0.357889	0.352351	0.346911	0.341567	0.336317	0.331159	0.32609
3	0.955620	0.942870	0.929761	0.916405	0.902892	0.889296	...	0.373555	0.367869	0.362283	0.356792	0.351396	0.346093	0.340880	0.33575
7	0.967152	0.959028	0.950934	0.942869	0.934834	0.926828	...	0.561230	0.555839	0.550497	0.545202	0.539956	0.534757	0.529605	0.52450
3	0.920782	0.898406	0.875956	0.853676	0.831726	0.810209	...	0.236111	0.231513	0.227024	0.222639	0.218356	0.214171	0.210084	0.20608
0	0.960245	0.949137	0.937733	0.926110	0.914328	0.902440	...	0.413868	0.408046	0.402314	0.396671	0.391115	0.385646	0.380260	0.37495
.	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
4	0.895196	0.867318	0.839884	0.813132	0.787198	0.762150	...	0.187415	0.183429	0.179547	0.175765	0.172079	0.168488	0.164988	0.16157
7	0.884526	0.854605	0.825374	0.797056	0.769766	0.743549	...	0.172052	0.168286	0.164620	0.161053	0.157579	0.154197	0.150903	0.14769
2	0.966311	0.957738	0.949110	0.940436	0.931725	0.922986	...	0.518455	0.512743	0.507093	0.501505	0.495977	0.490511	0.485104	0.47975
9	0.962449	0.952190	0.941692	0.931008	0.920183	0.909257	...	0.440037	0.434174	0.428396	0.422700	0.417087	0.411554	0.406100	0.40072
0	0.948374	0.933304	0.917854	0.902183	0.886413	0.870636	...	0.329600	0.324161	0.318826	0.313594	0.308461	0.303426	0.298486	0.29364

```
raw_data["CLV"] = clv_data.CLV
```

From these results we can understand that the customers who have higher CLV are the ones with low Churn risk.

```
print(raw_data.groupby(["gender", "ed","marital"])[["CLV"]].mean())
```

```
print(raw_data.groupby("gender")[["CLV"]].mean())
print(raw_data.groupby("voice")[["CLV"]].mean())
print(raw_data.groupby("forward")[["CLV"]].mean())
print(raw_data.groupby("internet")[["CLV"]].mean())
print(raw_data.groupby("marital")[["CLV"]].mean())
print(raw_data.groupby("region")[["CLV"]].mean())
print(raw_data.groupby("custcat")[["CLV"]].mean())
print(raw_data.groupby("retire")[["CLV"]].mean())
print(raw_data.groupby("ed")[["CLV"]].mean())
```

```
                                                          CLV
gender ed                           marital
Female College degree               Married    40060.421398
                                    Unmarried  37290.898426
       Did not complete high school Married    44650.213932
                                    Unmarried  44131.337060
       High school degree           Married    43580.198686
                                    Unmarried  40272.182004
       Post-undergraduate degree    Married    43213.370083
                                    Unmarried  33874.065001
       Some college                 Married    41179.712590
                                    Unmarried  40280.274795
Male   College degree               Married    42101.085262
                                    Unmarried  34854.913299
       Did not complete high school Married    48025.418050
                                    Unmarried  43284.201674
       High school degree           Married    46142.087572
                                    Unmarried  39360.851731
       Post-undergraduate degree    Married    41383.311367
                                    Unmarried  33144.101772
       Some college                 Married    44214.553963
                                    Unmarried  35335.748930
                 CLV
gender
Female  41126.506961
Male    41326.642952
                CLV
voice
No     42575.461690
Yes    38127.142462
                  CLV
forward
No       39698.658898
Yes      42790.978870
                   CLV
internet
No        44663.921886
Yes       35314.059816
                    CLV
marital
Married    43569.470093
Unmarried  38923.336531
                 CLV
region
Zone 1  41306.111183
Zone 2  41722.324987
Zone 3  40660.896214
                       CLV
custcat
Basic service  34882.570279
E-service      44558.848716
Plus service   46759.868046
Total service  38710.236686
                 CLV
retire
No      40703.694514
Yes     51756.420700
                                          CLV
```

The most noticeable difference in Customer Lifetime Value (CLV) is observed when considering the "retire" variable. This can be explained by the fact that older people are more conservative and tend to rely on the product that they are using. I could find a group with a high CLV. Those are the males who did not finish the high school and are married. In avarage they have around 48000 of CLV. I think that this has to do with the stability in their lives and the influence they might have on the surrounding people.

Conclusion

From our data we could understand that the higher is the CLV, the lower is the risk of churn.

The coefficients in our analysis carry specific implications:

- Positive coefficients signify that an increase in a given variable positively influences the anticipated customer lifetime.
- Negative coefficients indicate that an increase in a specific variable leads to a decrease in the expected customer lifetime.
- The magnitude of the coefficient reflects the strength of the variable's impact on customer lifetime.

For effective retention strategies based on CLV scores:

As the younger segment has lower CLV:

- Actively listen to customer feedback and address concerns promptly, particularly for the younger demographic.
- Implement exclusive discounts, special offers, or other perks for loyal customers, with a specific focus on the younger age group.
- Work on the internet quality, do better customer support for internet users, as the CLV of the people who use internet are low.

Budget

Taking arbitrary values for retention rate and cost per customer

```
dropped_data["CLV"] = clv_data.CLV


retained_customers = dropped_data[dropped_data['churn'] == 0]
retained_clv = retained_customers['CLV'].sum()


retention_rate = 0.8
cost_per_customer = 5000
retention_cost = len(dropped_data) * retention_rate * cost_per_customer


annual_budget = retained_clv - retention_cost
print("ANNUAL BUDGET:",annual_budget)
```

```
ANNUAL BUDGET: 27470982.86371857
```