AI5213/EC4213: Machine Learning and Deep Learning **Student ID: 20195024**
Homework 5 (SVM, Survival Analysis) **Name: Kim, Minjun**

Due: Sunday, Nov 19, 11:59PM

This homework comprises a set of conceptual problems and one coding exercise. Some problems are trivial, while others will require a lot of thought. Start this homework early!

> Guideline for those new to data analysis using Python:
> We recommend you to review the Lab section within each chapter (e.g., p. 387 of Ch. 9 or https://islp.readthedocs.io/en/latest/labs/Ch11-surv-lab.html) prior to tackling the programming tasks. Additionally, find datasets and Jupyter notebooks at https://github.com/intro-stat-learning/ISLP_labs/ and https://islp.readthedocs.io/en/latest/.
>
> Visit https://www.statlearning.com/forum — a dedicated forum created by and for the ISL community. Whether you have a question or encounter issues with ISLP labs, this platform is your go-to resource for assistance and collaborative discussions.

Deliverables:

1. Submit a PDF of your homework, with an appendix listing all your code, to the Gradescope assignment entitled "HW5 Write-Up". You may typeset your homework in LaTeX or Word or submit neatly handwritten and scanned solutions. Please start each question on a new page. If there are graphs, include those graphs in the correct sections. **Take a screenshot of the code and attach to each question.** Do not put them in an appendix. We need each solution to be self-contained on pages of its own.

   • On the first page of your write-up, please sign your signature next to the following statement. (Mac Preview, PDF Expert, and Foxit PDF Reader, among others, have tools to let you sign a PDF file.) We want to make extra clear the consequences of cheating.

   "I certify that all solutions are entirely in my own words and that I have not looked at another student's solutions. I have given credit to all external sources I consulted."

   • On the first page of your write-up, please list students who helped you or whom you helped on the homework. (Note that sending each other code is not allowed.)

2. Submit all the code needed to reproduce your results to the Gradescope assignment entitled "HW5 Code". You must submit your code twice: once in your PDF write-up (above) so the readers can easily read it, and again in compilable/interpretable form so the readers can easily run it. Do NOT include any data files we provided. Please include a short file named README listing your name, student ID, and instructions on how to reproduce your results. Please take care that your code doesn't take up inordinate amounts of time or memory.

For staff use only

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Total
/ 9	/ 12	/ 8	/ 12	/ 14	/ 15	/ 30	/ 100

Honor Code

Declare and sign the following statement:

"I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted."

Signature: I certify that all solutions in this document are entirely my own and that I have not looked at anyone else's solution. I have given credit to all external sources I consulted.

We welcome group discussions, but the work you submit should be entirely your own. If you use any information or pictures not from our lectures or readings, make sure to say where they came from. Please note that breaking academic rules can lead to severe penalties.

(a) Did you receive any help whatsoever from anyone in solving this assignment? If your answer is 'yes', give full details (e.g. "Junho explained to me what is asked in Q2-a")

No.

(b) Did you give any help whatsoever to anyone in solving this assignment? If your answer is 'yes', give full details (e.g. "I pointed Josh to Ch. 2.3 since he didn't know how to proceed with Q2")

No.

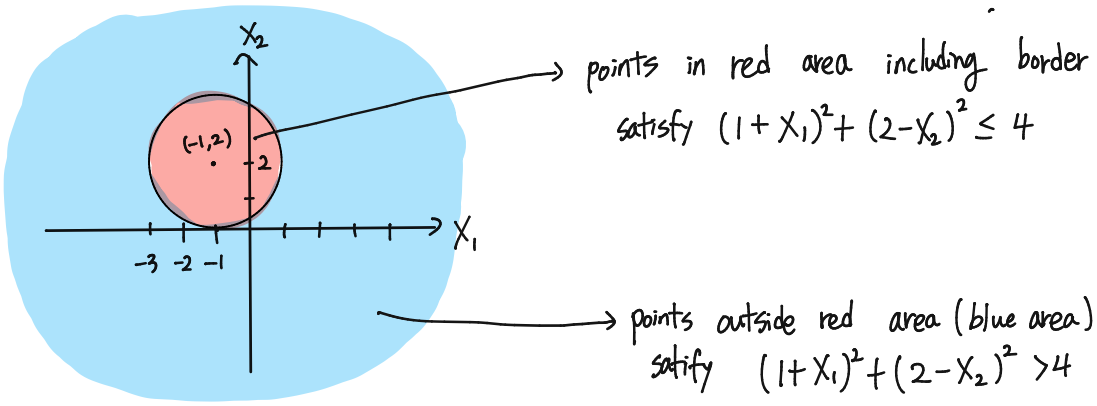
(c) Did you find or come across code that implements any part of this assignment? If your answer is 'yes', give full details (book & page, URL & location within the page, etc.).

I reviewed Lab session in Chapter 9 and 11, before I jumped into this assignment.

Q1. Non-linear decision boundary [9 pts]

We have seen that in $p = 2$ dimensions, a linear decision boundary takes the form $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$. We now investigate a non-linear decision boundary.

(a) Sketch the curve $(1 + X_1)^2 + (2 - X_2)^2 = 4$, and indicate the set of points for which $(1 + X_1)^2 + (2 - X_2)^2 > 4$ as well as the set of points for which $(1 + X_1)^2 + (2 - X_2)^2 \leq 4$. [3 pts]



(b) Suppose that a classifier assigns an observation to the blue class if $(1 + X_1)^2 + (2 - X_2)^2 > 4$, and to the red class otherwise. To what class is the observation $(0, 0)$ classified? $(-1, 1)$? $(2, 2)$? $(3, 8)$? [3 pts]

$(0,0)$: $1^2+2^2=5$, $\therefore$ blue class

$(-1,1)$: $0^2+1^2=1$, $\therefore$ red class

$(2,2)$: $3^2+0^2=9$, $\therefore$ blue class

$(3,8)$: $4^2+6^2=80$, $\therefore$ blue class

(c) Argue that while the decision boundary in ~~(c)~~ (b) is not enough in terms of X_1 and X_2, it is linear in terms of X_1, X_1^2, X_2, and X_2^2. [3 pts]

The decision boundary equation is $(1+X_1)^2+(2-X_2)^2=4$.

That is $1+2X_1+X_1^2-2X_2+X_2^2=0$.

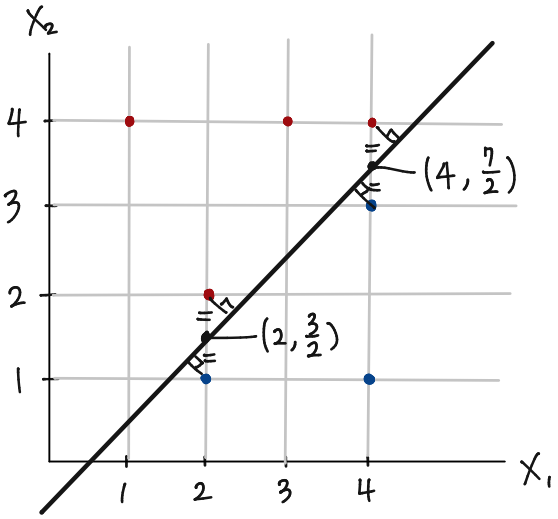
So it takes a linear form in terms of X_1, X_1^2, X_2, and X_2^2.

Q2. Maximal margin classifier [12 pts]

Here we explore the maximal margin classifier on a toy data set.

(a) We are given $n = 7$ observations in $p = 2$ dimensions. For each observation, there is an associated class label. Sketch the observations and the optimal separating hyperplane. Provide the equation for this hyperplane (of the form similar to $\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$). [3 pts]

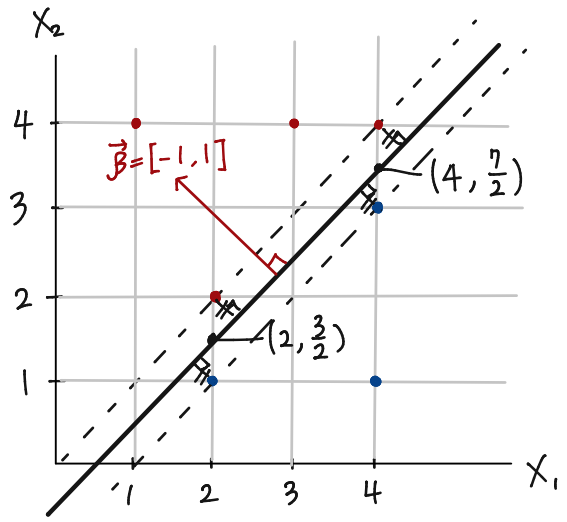
Obs	X_1	X_2	Y
1	3	4	Red
2	2	2	Red
3	4	4	Red
4	1	4	Red
5	2	1	Blue
6	4	3	Blue
7	4	1	Blue



I chose the hyperplane passing through $(2, \frac{3}{2})$ and $(4, \frac{7}{2})$ like a graph above.

The equation is, $X_2 - \frac{3}{2} = \frac{\frac{7}{2} - \frac{3}{2}}{4-2}(X_1 - 2)$, which is $\frac{1}{2} - X_1 + X_2 = 0$.

(b) Describe the classification rule for the maximal margin classifier. It should be something along the lines of "Classify to Red if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > 0$, and classify to Blue otherwise." Provide the values for β_0, β_1, and β_2. [2 pts]



The hyperplane of problem (a) is already satisfying the maximal margin classifier, since the distance between the closest red point and the hyperplane is the same as the distance between the closest blue point and the hyperplane.

The normal vector $\vec{\beta}$ is $[-1, 1]$.

Therefore, if $f(X) = \frac{1}{2} - X_1 + X_2$,

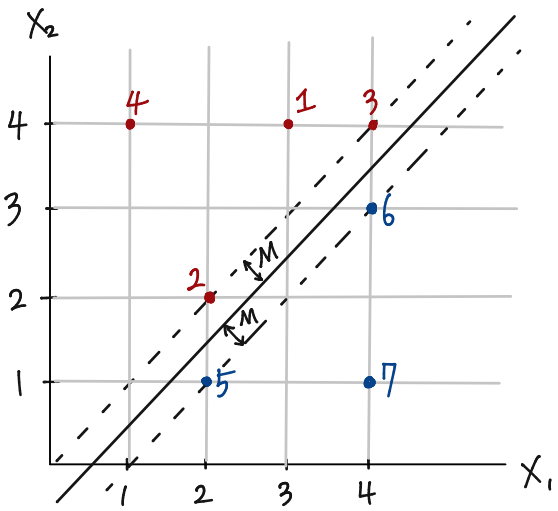
then $f(X) > 0$ for Red points, $f(X) < 0$ for Blue points.

$f(X) = 0$ defines a seperating hyperplane.

If we consider the condition $\sum_{j=1}^{p} \beta_j^2 = 1$, then $s^2(\frac{1}{4} + 1 + 1) = 1$ where s is scaling factor.

$s = \frac{2}{3}$, then $f(X) = \frac{1}{3} - \frac{2}{3}X_1 + \frac{2}{3}X_2$. $\therefore \beta_0 = \frac{1}{3}$, $\beta_1 = -\frac{2}{3}$, $\beta_2 = \frac{2}{3}$.

(c) On your sketch, indicate the margin for the maximal margin hyperplane. Also, indicate the support vectors for the maximal margin classifier. Next, argue that a slight movement of the seventh observation would not affect the maximal margin hyperplane. [3 pts]

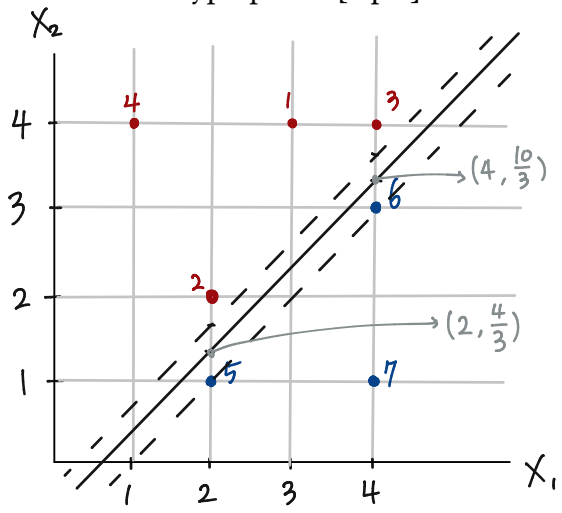


I've already indicated margin in the previous graph in problem (b).
I assigned numbers to each point to indicate indices of observation.
Then, 2, 3, 5 and 6 are support vectors, which influences the formation of hyperplane and maximize the margin.

$f(X) = \frac{1}{3} - \frac{2}{3}X_1 + \frac{2}{3}X_2 \quad \therefore M = \frac{\left|-\frac{1}{3} - \frac{2}{3}\cdot 2 + \frac{2}{3}\cdot 2\right|}{\sqrt{\left(-\frac{2}{3}\right)^2 + \left(\frac{2}{3}\right)^2}} = \frac{\frac{1}{3}}{\frac{2\sqrt{2}}{3}} = \frac{\sqrt{2}}{4}$

A slight movement of the seventh observation would not affect the maximal margin classifier, because it is not support vector and I didn't consider 7 when I built this SVM. However, if 7 move inside the current margin, both hyperplane and margin will be changed.

(d) Sketch a hyperplane that is *not* the optimal separating hyperplane, and provide the equation for this hyperplane. [2 pts]

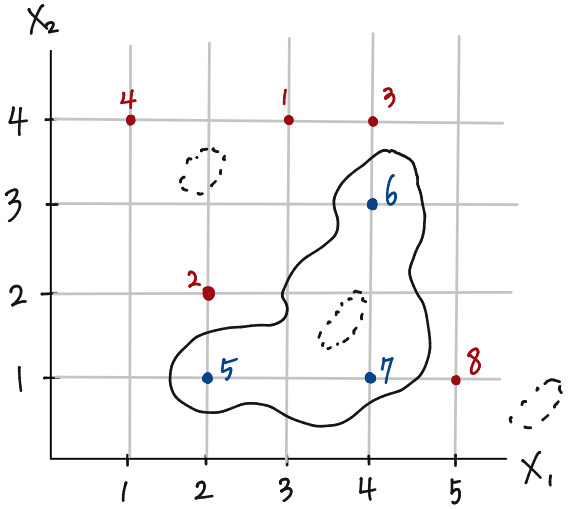


I chose a hyperplane which passes through $(4, \frac{10}{3})$ and $(2, \frac{4}{3})$.
The equation is $X_2 - \frac{4}{3} = \frac{\frac{10}{3} - \frac{4}{3}}{4-2}(X_1 - 2)$, which is

$$\frac{2}{3} - X_1 + X_2 = 0.$$

This is not optimal because this classifier does not maximize the margin.
Now the margin M' is $M' = \frac{\left|\frac{2}{3} - 2 + 1\right|}{\sqrt{2}} = \frac{\sqrt{2}}{6}$

(e) Draw an additional observation on the plot so that the two classes are no longer separable by a hyperplane. [2 pts]



I added 8th observation on (5, 1), of which Y is Red.
Then, it is impossible to seperate the two classes by a hyperplane in 2-dimensional space.
Therefore we should enlarge the dimension of feature space.
For example, I can use kernal functions to deal with this issue.

Q3. Independence of the cersoring mechanism [8 pts]

For each example, state whether or not the censoring mechanism is independent. Justify your answer.

(a) In a study of disease relapse, due to a careless research scientist, all patients whose phone numbers begin with the number "2" are lost to follow up. [2 pts]

In this problem, the censoring mechanism is independent. This is because the phone number starting with "2" is completely random, so it is not related to the patient's disease relapse.

However, if the phone number starting with "2" is related to the area where the patients live(for example 02 for Seoul and 051 for Busan in Korea), the censoring mechanism may not be independent because environmental and genetic factors in the area can affect the disease relapse.

(b) In a study of longevity, a formatting error causes all patient ages that exceed 99 years to be lost (i.e. we know that those patients are more than 99 years old, but we do not know their exact ages). [2 pts]

In this problem, the censoring mechanism is not independent. Since the purpose of this study is about longevity, the age of patients over 99 years is directly related to longevity. Therefore, censoring can have a significant impact on the results of the study.

Moreover, as we do not know the 'exact' ages of patients more than 99 years, patients who can be classified as 'over 105 years old', or 'over 110 years old', are all classified as just 'over 99 years old', which can lead to bias problem.

(c) Hospital A conducts a study of longevity. However, very sick patients tend to be transferred to Hospital B, and are lost to follow up. [2 pts]

In this problem, the censoring mechanism is not independent. The transfer of critically ill patients from Hospital A to Hospital B is completely related to health status of patients, which significantly affects their longevity. Therefore, this violate the assumption of independent censoring.

(d) In a study of unemployment duration, the people who find work earlier are less motivated to stay in touch with study investigators, and therefore are more likely to be lost to follow up. [2 pts]

In this problem, the censoring mechanism is not independent. If people who find work earlier are censored from the study, the people who remain in the study are more likely to experience longer periods of unemployment. This suggests that the study may overestimate the true duration of unemployment. Therefore, this violate the assumption of independent censoring.

Q4. Survival curve [12 pts]

This problem makes use of the data displayed in Figure 1. In completing this problem, you can refer to the observation times as $y_1, \cdots, y_4$. The ordering of these observation times can be seen from Figure 1; their exact values are not required.

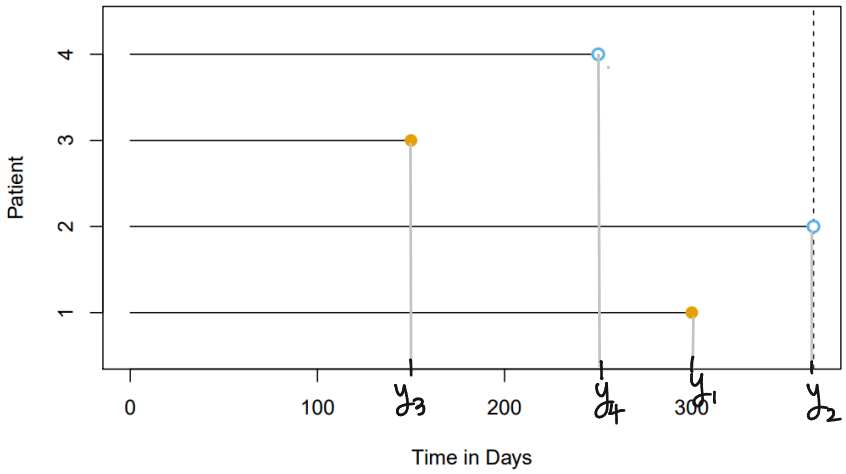


Figure 1: An illustration of censored survival data from a medical study in which we observe $n = 4$ patients for a 365-day follow-up period. For patients 1 and 3, the event was observed. Patient 2 was alive when the study ended. Patient 4 dropped out of the study.

(a) Report the values of $\delta_1, \cdots, \delta_4, K, d_1, \cdots, d_K, r_1, \cdots, r_K$, and $q_1, \cdots, q_K$. The relevant notation is defined in Sections 11.1 and 11.3 of the ISLP book. [3 pts]

$$\delta = \begin{cases} 1 & \text{if } T \le C \\ 0 & \text{if } T > C \end{cases}, \quad \therefore \delta_1 = \delta_3 = 1, \ \delta_2 = \delta_4 = 0.$$

$K = 2$

$d_1 < d_2$ are the unique death times among the non-censored patients.

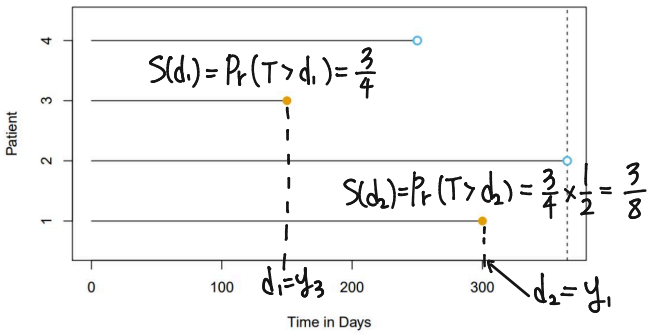
r_k is the number of patients at risk at time d_k.

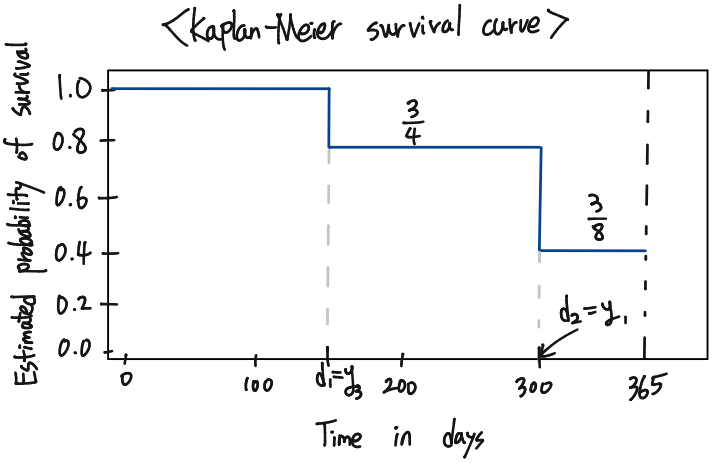
q_k is the number of patients who died at time d_k.

$\therefore d_1 = y_3, \ r_1 = 4, \ q_1 = 1$

$d_2 = y_1, \ r_2 = 2, \ q_2 = 1$

(b) Sketch the Kaplan-Meier survival curve corresponding to this data set. Do not use any software to do this — you can sketch it by hand using the results obtained in (a). Based on the survival curve, what is the probability that the event occurs within 200 days? What is the probability that the event does not occur within 310 days? [6 pts]





i) The probability that the events occurs with in 200 days is $1 - \frac{3}{4} = \frac{1}{4}$.

ii) The probability that the probability that the event does not occur within 310 day is $\frac{3}{8}$.

(c) Write out an expression for the estimated survival curve from (b). [3 pts]

I've already shown an expression in the above problem, but I'll write down one more time.

Let's remind that $S(d_k) = \prod_{j=1}^{k} \left(\frac{r_j - q_j}{r_j} \right)$, which is the probability that a patient survives for at least $t = d_k$ days.

$$\therefore \quad S(d_1) = \frac{r_1 - q_1}{r_1} = \frac{4-1}{4} = \frac{3}{4}.$$

$$S(d_2) = \frac{r_1 - q_1}{r_1} \cdot \frac{r_2 - q_2}{r_2} = \frac{3}{4} \cdot \frac{2-1}{2} = \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{8}$$

$$\therefore \text{ Estimated probability of survival } S(t) = \begin{cases} 1, & 0 < t < d_1 \\ \frac{3}{4}, & d_1 \le t < d_2 \\ \frac{3}{8}, & d_2 \le t < 365 \end{cases}$$

Q5. Support Vector Machine [14 pts]

In this problem, you will use support vector approaches in order to predict whether a given car gets high or low aagas mileage based on the Auto data set. Make some plots to back up your assertions in (a) and (b).

Hint: In the lab, we used the plot_svm() function for fitted SVMs. When $p > 2$, you can use the keyword argument features to create plots displaying pairs of variable at a time.

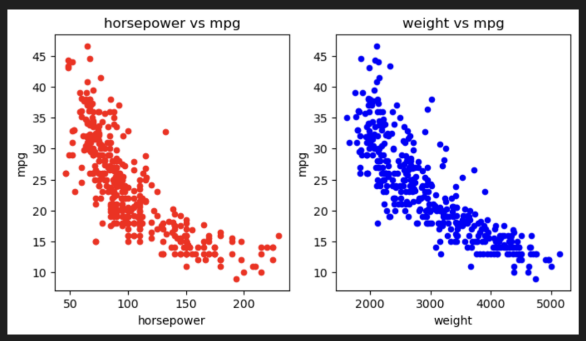
(a) Create a binary variable that takes on a 1 for cars with gas mileage above the median, and a 0 for cars with gas mileage below the median. Fit a support vector classifier to the data with various values of C, in order to predict whether a car gets high or low gas mileage. Report the cross-validation errors associated with different values of this parameter. Comment on your results. Note you will need to fit the classifier without the gas mileage variable to produce sensible results. [7 pts]

```
#Set 1 for cars with gas mileage above the median, and 0 for cars with gas mileage below the median

median_mpg=Auto['mpg'].median()
print(f"Median of gas mileage is {median_mpg}")
Auto['binary'] = (Auto['mpg'] > median_mpg).astype(int)
```
[38] Python

```
Median of gas mileage is 22.75
```

```
fig, axes=plt.subplots(ncols=2, figsize=(8, 4))
Auto.plot.scatter('horsepower', 'mpg', ax=axes[0],color='red')
Auto.plot.scatter('weight', 'mpg', ax=axes[1], color='blue')
axes[0].set_title("horsepower vs mpg")
axes[1].set_title("weight vs mpg")
```
[39] Python

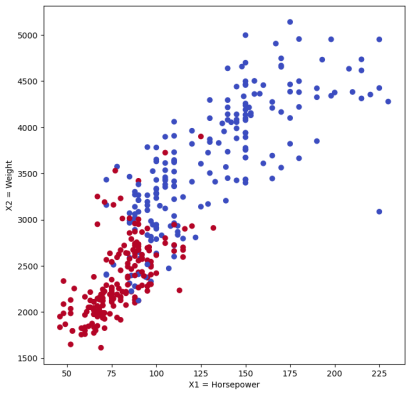
```
Text(0.5, 1.0, 'weight vs mpg')
```



Through this, I decided to set X1 as horsepower and X2 as weight in SVM classifier graph, because they seem to have a linear relationship with mpg. I thought the SVM classifier would work well.

```python
#Make SVM classifier plots

X=Auto[['horsepower', 'weight']].to_numpy()
y=Auto['binary'].to_numpy()

fig, ax=plt.subplots(figsize=(8,8))
scatter=ax.scatter(X[:,0], X[:,1], c=y, cmap='coolwarm')
plt.xlabel('X1 = Horsepower')
plt.ylabel('X2 = Weight')
plt.show()
```

[84] Python

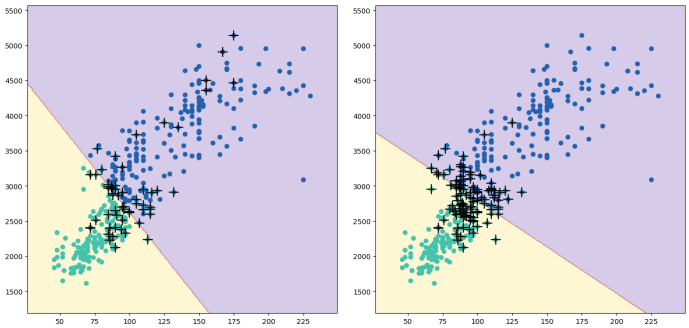



```python
#cost parameter = 10
svm_linear=SVC(C=10, kernel='linear')
svm_linear.fit(X, y)
fig, ax=subplots(figsize=(8,8))
plot_svm(X, y, svm_linear, ax=ax)
```

[90] Python

```python
#cost parameter=0.1
svm_linear=SVC(C=0.1, kernel='linear')
svm_linear.fit(X, y)
fig, ax=subplots(figsize=(8,8))
plot_svm(X, y, svm_linear, ax=ax)
```

[91] Python



A lower cost parameter results in an increased number of support vectors due to the expansion of the margin.

```
cost_values=[0.001, 0.01, 0.1, 1, 5, 10, 100]

kfold=skm.KFold(5, random_state=0, shuffle=True)
grid=skm.GridSearchCV(svm_linear, {'C':cost_values}, refit=True, cv=kfold, scoring='accuracy')
grid.fit(X, y)
scores=grid.cv_results_[('mean_test_score')]
```

[133] Python

```
for i, cost in enumerate(cost_values):
    print(f"Cost parameter = {cost} : Cross-validation accuracy = {scores[i]:.4f}")


optimal_index = np.argmax(scores)
best_cost = cost_values[optimal_index]
print(f"\nOptimal cost parameter : {best_cost}")
```

[110] Python

```
Cost parameter = 0.001 : Cross-validation accuracy = 0.8726
Cost parameter = 0.01 : Cross-validation accuracy = 0.8751
Cost parameter = 0.1 : Cross-validation accuracy = 0.8751
Cost parameter = 1 : Cross-validation accuracy = 0.8751
Cost parameter = 5 : Cross-validation accuracy = 0.8929
Cost parameter = 10 : Cross-validation accuracy = 0.8853
Cost parameter = 100 : Cross-validation accuracy = 0.8802

Optimal cost parameter : 5
```

I found that when cost parameter is 5, then the cross-validation accuracy is the highest.

(b) Now repeat (a), this time using SVMs with radial and polynomial basis kernels, with different values of gamma and degree and C. Comment on your results. [7 pts]

```
[162]
X=Auto[['horsepower', 'weight']].to_numpy()
y=Auto['binary'].to_numpy()
```
Python

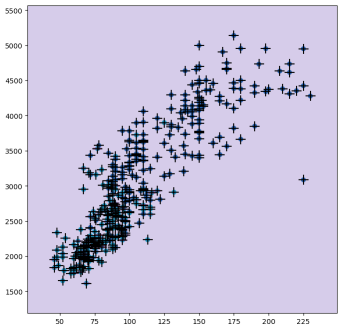
```
[163]
#radial basis kernal
cost_values=[0.001, 0.01, 0.1, 1, 5, 10, 100]
gamma_values=[0.5, 1, 2, 3, 4]

svm_radial=SVC(kernel="rbf")
kfold=skm.KFold(5, random_state=0, shuffle=True)
grid=skm.GridSearchCV(svm_radial, {'C':cost_values, 'gamma':gamma_values}, refit=True, cv=kfold, scoring='accuracy')
grid.fit(X, y)
scores=grid.cv_results_[('mean_test_score')]

print(f"Optimal cost parameter and gamma value : {grid.best_params_}")
print(f"Best Cross-validation Accuracy : {grid.best_score_:.4f}")
```
Python

```
Optimal cost parameter and gamma value : {'C': 5, 'gamma': 0.5}
Best Cross-validation Accuracy : 0.5125
```

```
[164]
for i, cost in enumerate(cost_values):
    for j, gamma in enumerate(gamma_values):
        score_index = i * len(gamma_values) + j
        print(f"Cost = {cost}, Gamma = {gamma}: Cross-validation accuracy = {scores[score_index]:.4f}")
```
Python

```
Cost = 0.001, Gamma = 0.5: Cross-validation accuracy = 0.4311
Cost = 0.001, Gamma = 1: Cross-validation accuracy = 0.4311
Cost = 0.001, Gamma = 2: Cross-validation accuracy = 0.4311
Cost = 0.001, Gamma = 3: Cross-validation accuracy = 0.4311
Cost = 0.001, Gamma = 4: Cross-validation accuracy = 0.4311
Cost = 0.01, Gamma = 0.5: Cross-validation accuracy = 0.4311
Cost = 0.01, Gamma = 1: Cross-validation accuracy = 0.4311
Cost = 0.01, Gamma = 2: Cross-validation accuracy = 0.4311
Cost = 0.01, Gamma = 3: Cross-validation accuracy = 0.4311
Cost = 0.01, Gamma = 4: Cross-validation accuracy = 0.4311
Cost = 0.1, Gamma = 0.5: Cross-validation accuracy = 0.4311
Cost = 0.1, Gamma = 1: Cross-validation accuracy = 0.4311
Cost = 0.1, Gamma = 2: Cross-validation accuracy = 0.4311
Cost = 0.1, Gamma = 3: Cross-validation accuracy = 0.4311
Cost = 0.1, Gamma = 4: Cross-validation accuracy = 0.4311
Cost = 1, Gamma = 0.5: Cross-validation accuracy = 0.5100
Cost = 1, Gamma = 1: Cross-validation accuracy = 0.5074
Cost = 1, Gamma = 2: Cross-validation accuracy = 0.4821
Cost = 1, Gamma = 3: Cross-validation accuracy = 0.4668
Cost = 1, Gamma = 4: Cross-validation accuracy = 0.4617
Cost = 5, Gamma = 0.5: Cross-validation accuracy = 0.5125
Cost = 5, Gamma = 1: Cross-validation accuracy = 0.5074
Cost = 5, Gamma = 2: Cross-validation accuracy = 0.4821
Cost = 5, Gamma = 3: Cross-validation accuracy = 0.4745
Cost = 5, Gamma = 4: Cross-validation accuracy = 0.4617
...
Cost = 100, Gamma = 1: Cross-validation accuracy = 0.5074
Cost = 100, Gamma = 2: Cross-validation accuracy = 0.4821
Cost = 100, Gamma = 3: Cross-validation accuracy = 0.4745
Cost = 100, Gamma = 4: Cross-validation accuracy = 0.4617
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

```
[155]
#This performs much worse than the linear SVM in problem(a), because the data is not gathered in a circle.

svm_radial=SVC(kernel="rbf", gamma=0.5 ,C=5)
svm_radial.fit(X,y)
fig, ax=subplots(figsize=(8,8))
plot_svm(X, y, svm_radial, ax=ax)
```
Python



I found that optimal parameters are 'C': 5, 'gamma': 0.5. But the cross-validation accuracy was much worse than those of problem (a).

```python
#polynomial basis kernal
cost_values=[0.001, 0.01, 0.1, 1, 5, 10, 100]
degree_values=[2, 3, 4, 5]

svm_polynomial=SVC(kernel="poly")
kfold=skm.KFold(5, random_state=0, shuffle=True)
grid=skm.GridSearchCV(svm_polynomial, {'C':cost_values, 'degree':degree_values}, refit=True, cv=kfold, scoring='accuracy')
grid.fit(X, y)
scores=grid.cv_results_[('mean_test_score')]

print(f"Optimal cost parameter and gamma value : {grid.best_params_}")
print(f"Best Cross-validation Accuracy : {grid.best_score_:.4f}")
```

[165] Python

```
Optimal cost parameter and gamma value : {'C': 100, 'degree': 4}
Best Cross-validation Accuracy : 0.8853
```

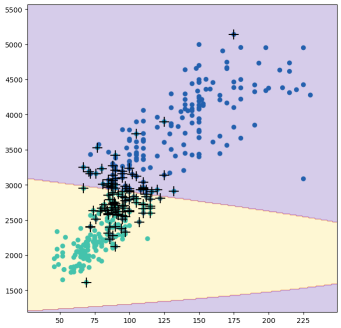
```python
for i, cost in enumerate(cost_values):
    for j, degree in enumerate(degree_values):
        score_index = i * len(degree_values) + j
        print(f"Cost = {cost}, Degree = {degree}: Cross-validation accuracy = {scores[score_index]:.4f}")
```

[166] Python

```
Cost = 0.001, Degree = 2: Cross-validation accuracy = 0.4695
Cost = 0.001, Degree = 3: Cross-validation accuracy = 0.7575
Cost = 0.001, Degree = 4: Cross-validation accuracy = 0.8264
Cost = 0.001, Degree = 5: Cross-validation accuracy = 0.8494
Cost = 0.01, Degree = 2: Cross-validation accuracy = 0.8468
Cost = 0.01, Degree = 3: Cross-validation accuracy = 0.8673
Cost = 0.01, Degree = 4: Cross-validation accuracy = 0.8724
Cost = 0.01, Degree = 5: Cross-validation accuracy = 0.8674
Cost = 0.1, Degree = 2: Cross-validation accuracy = 0.8726
Cost = 0.1, Degree = 3: Cross-validation accuracy = 0.8726
Cost = 0.1, Degree = 4: Cross-validation accuracy = 0.8751
Cost = 0.1, Degree = 5: Cross-validation accuracy = 0.8700
Cost = 1, Degree = 2: Cross-validation accuracy = 0.8803
Cost = 1, Degree = 3: Cross-validation accuracy = 0.8700
Cost = 1, Degree = 4: Cross-validation accuracy = 0.8751
Cost = 1, Degree = 5: Cross-validation accuracy = 0.8725
Cost = 5, Degree = 2: Cross-validation accuracy = 0.8803
Cost = 5, Degree = 3: Cross-validation accuracy = 0.8803
Cost = 5, Degree = 4: Cross-validation accuracy = 0.8751
Cost = 5, Degree = 5: Cross-validation accuracy = 0.8674
Cost = 10, Degree = 2: Cross-validation accuracy = 0.8803
Cost = 10, Degree = 3: Cross-validation accuracy = 0.8751
Cost = 10, Degree = 4: Cross-validation accuracy = 0.8726
Cost = 10, Degree = 5: Cross-validation accuracy = 0.8598
Cost = 100, Degree = 2: Cross-validation accuracy = 0.8803
Cost = 100, Degree = 3: Cross-validation accuracy = 0.8828
Cost = 100, Degree = 4: Cross-validation accuracy = 0.8853
Cost = 100, Degree = 5: Cross-validation accuracy = 0.8751
```

```python
#This performs a little worse than the linear SVM in problem(a), because the data are linearly listed.

svm_polynomial=SVC(kernel="rbf", C=100, degree=4)
svm_polynomial.fit(X,y)
fig, ax=subplots(figsize=(8,8))
plot_svm(X, y, svm_polynomial, ax=ax)
```

[168] Python



I found that optimal parameters are 'C': 100, 'degree': 4. The cross-validation accuracy was a little worse than those of problem (a).

Q6. Survival analysis on brain tumor data [15 pts]

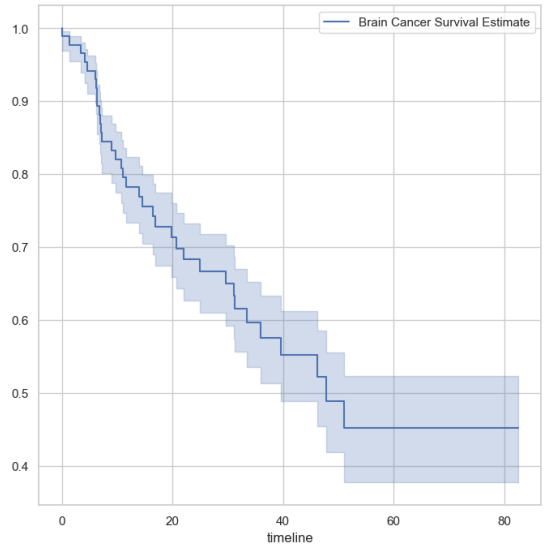
This exercise focuses on the brain tumor data, which is included in the ISLP library.

(a) Plot the Kaplan-Meier survival curve with ± 1 standard error bands, using the KaplanMeierFitter() estimator in the lifeline package. [3 pts]

The "±1 standard error band" is essentially similar to a 68% confidence interval. In a normal distribution, values within ±1 standard error of the estimate include approximately 68% of the total values.

```
#Therefore I should set alpha=1-0.68=0.32
fig, ax = plt.subplots(figsize=(8, 8))
km = KaplanMeierFitter(alpha=0.32)
km_brain = km.fit(BrainCancer['time'],BrainCancer['status'])
km_brain.plot(label='Brain Cancer Survival Estimate',ax=ax)
```

[66] ✓ 0.1s Python




```
#I calculated the standard error for Kaplan-Meier survival estimates at each time point
se = km.confidence_interval_.iloc[:, 1] - km.survival_function_.iloc[:, 0]
se
```

[67] ✓ 0.0s Python

```
0.00     0.000000
0.07     0.007145
1.18     0.007145
1.41     0.011478
1.54     0.011478
           ...
65.02    0.071286
67.38    0.071286
73.74    0.071286
78.75    0.071286
82.56    0.071286
Length: 87, dtype: float64
```

(b) Draw a bootstrap sample of size $n = 88$ from the pairs (y_i, δ_i), and compute the resulting Kaplan-Meier survival curve. Repeat this process $B = 200$ times. Use the results to obtain an estimate of the standard error of the Kaplan-Meier survival curve at each timepoint. Compare this to the standard errors obtained in (a). [5 pts]

```python
#There are 86 unique timepoint in BrainCancer data set.
unique_timepoints=BrainCancer['time'].unique()
print(len(unique_timepoints))
```

[68] ✓ 0.0s Python

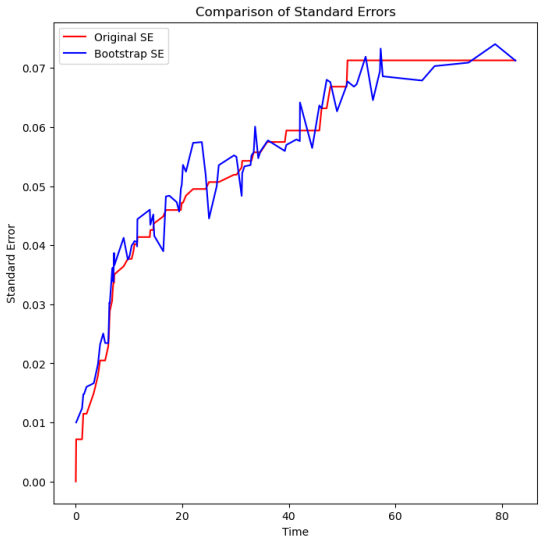
```
86
```

```python
n=88 #size of each boostrap sample
B=200 #the number of repeating times
unique_timepoints = np.sort(unique_timepoints)  #sorting in ascending order

#I made an array to store Kaplan-Meier estimates for each bootstrap sample
km_estimates=np.zeros((B, len(unique_timepoints)))

for i in range(B):
    #generate bootstrap sample
    bootstrap_indices=np.random.choice(range(n), size=n, replace=True)
    bootstrap_sample=BrainCancer.iloc[bootstrap_indices]

    km_bootstrap=KaplanMeierFitter()
    km_bootstrap.fit(bootstrap_sample['time'], bootstrap_sample['status'])

    #store survival estimates at each timepoint
    for j, timepoint in enumerate(unique_timepoints):
        if timepoint in km_bootstrap.survival_function_.index:
            km_estimates[i, j]=km_bootstrap.survival_function_.loc[timepoint]

        else:  #this is a case if timepoint is not in survival function
            km_estimates[i, j]=np.nan

#calculate standard error at each timepoint
se_bootstrap=np.nanstd(km_estimates, axis=0)

for timepoint, error in zip(unique_timepoints, se_bootstrap):
    print(f"Timepoint: {timepoint}, SE: {error}")
```

[69] Python

```python
plt.figure(figsize=(8, 8))
plt.plot(km.survival_function_.index, se, label='Original SE', color='red')
plt.plot(unique_timepoints, se_bootstrap, label='Bootstrap SE', color='blue')

plt.xlabel('Time')
plt.ylabel('Standard Error')
plt.title('Comparison of Standard Errors')
plt.legend()
plt.show()
```

[70] Python



I could see there are similarities between the standard errors calculated using the original Kaplan-Meier method and the bootstrap approach. This indicates that the bootstrap approach effectively captures the variability of the Kaplan-Meier estimates from the original dataset.

(c) Fit a Cox proportional hazards model that uses all of the predictors to predict survival. Summarize the main findings. [3 pts]

```
#I found that there are some category type variables.
BrainCancer = load_data('BrainCancer')
BrainCancer.dtypes
```

[64] ✓ 0.0s Python

```
sex          category
diagnosis    category
loc          category
ki              int64
gtv           float64
stereo       category
status          int64
time          float64
dtype: object
```

```
#I used ModelSpec and fit_transform, I could deal with categorical variables.
#And there are NaN values in diagnosis columns, so I dropped them.
processed_BrainCancer=BrainCancer.dropna()
model_df = MS(processed_BrainCancer.columns, intercept=False).fit_transform(processed_BrainCancer)
coxph = CoxPHFitter
cox_fit = coxph().fit(model_df,'time','status')
cox_fit.summary[['coef', 'se(coef)', 'p']]
```

[69] ✓ 0.0s Python

covariate	coef	se(coef)	p
sex[Male]	0.183748	0.360358	0.610119
diagnosis[LG glioma]	-1.239530	0.579555	0.032455
diagnosis[Meningioma]	-2.154566	0.450524	0.000002
diagnosis[Other]	-1.268870	0.617672	0.039949
loc[Supratentorial]	0.441195	0.703669	0.530665
ki	-0.054955	0.018314	0.002693
gtv	0.034293	0.022333	0.124661
stereo[SRT]	0.177778	0.601578	0.767597

```
processed_BrainCancer['diagnosis'].unique()
```

[70] ✓ 0.0s Python

```
['Meningioma', 'HG glioma', 'LG glioma', 'Other']
Categories (4, object): ['HG glioma', 'LG glioma', 'Meningioma', 'Other']
```

If we look at the table above, we see that the standard group is 'HG glinoma', because there are only diagnosis[LG glioma], diagnosis [Meningioma], and diagnosis[Other] in the table.

I focused on the results with a p-value less than 0.05. Since the coefficient of diagnosis[Meningioma] is -2.154566, this indicates that patients diagnosed with Meningioma have a statistically significantly lower risk compared to the baseline diagnosis, the HG glioma.

Likewise, the coefficient of diagnosis[LG glioma] is -1.239530, so patients with LG glioma have a lower risk compared to the those with HG glioma.

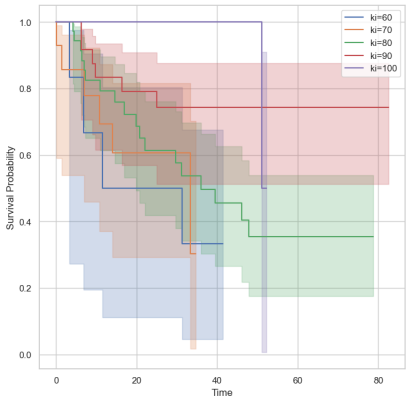
Finally, ki, the Karnofsky index, of which the coefficient is -0.054955, is associated with a decrease in the risk.

(d) Stratify the data by the value of ki. (Since only one observation has ki==40, you can group that observation together with the observations that have ki==60. Plot Kaplan-Meier survival curves for each of the five strata, adjusted for the other predictors. [4 pts]

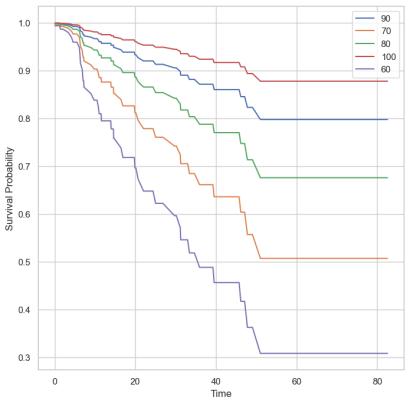
```
processed_BrainCancer['ki'].unique()
```
[71] ✓ 0.0s Python

```
array([ 90,  70,  80, 100,  60,  40])
```

```
#Since there are only one obesevation has ki=40,, I group that together with ki=60
processed_BrainCancer['ki_grouped'] = processed_BrainCancer['ki'].replace(40, 60)

fig, ax = plt.subplots(figsize=(8, 8))
km = KaplanMeierFitter()

#I drew Kaplan-Meier curve for each kj group.
for ki, df in processed_BrainCancer.groupby('ki_grouped'):
    km.fit(df['time'], df['status'])
    km.plot(label=f'ki={ki}', ax=ax)

plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.legend()
plt.show()
```
[55] ✓ 0.1s Python




```
#I revised the survival curves using the Cox model to adjust for other predictors.

ki_levels=processed_BrainCancer['ki'].replace(40, 60).unique()
def representative(series):
    if hasattr(series.dtype, 'categories'):
        return pd.Series.mode(series) #최빈값
    else:
        return series.mean() #평균
representative_data = processed_BrainCancer.apply(representative, axis=0)

#I created a DataFrame with repeated rows of representative values for each 'ki' level.
model_df = pd.DataFrame([representative_data.iloc[0] for _ in range(len(ki_levels))])

#I assigned the 'ki' ki_levels to the new DataFrame.
model_df['ki'] = ki_levels

model_spec = MS(processed_BrainCancer.columns, intercept=False)
transformed_df = model_spec.fit_transform(processed_BrainCancer)
ki_transformed_X = model_spec.transform(model_df)
ki_transformed_X.index = ki_levels
predicted_survival = cox_fit.predict_survival_function(ki_transformed_X)

fig, ax = subplots(figsize=(8, 8))
predicted_survival.plot(ax=ax)
plt.xlabel('Time')
plt.ylabel('Survival Probability')
plt.legend()
plt.show()
```
[81] ✓ 0.2s Python



Q7. Essay: Predicting submission behavior in Kaggle competitions [30 pts]

Background: In Kaggle competitions, participants submit their solutions to problems and receive scores based on the quality of their solutions. Submission behavior varies among participants: while some may submit only once, others might submit multiple times in pursuit of a better score. This behavior can be influenced by various factors, including the participant's academic background, academic workload, intellectual capacity, personality, and their current standing in the competition.

Objective: Using the Hawkes process and survival analysis, develop a survival regression model to forecast if and when a student will make a subsequent submission to a Kaggle competition leaderboard (Hawkes process is a kind of self-exciting point process, see Figure 2 for illustration).

Datasets Provided:

- *UserProfile* table (Table 1): Contains personal and academic information for each student.
- *SubmissionRecord* table (Table 2): Contains time series data for each student's submission, including the timestamp, accuracy, and whether the target accuracy was achieved. The initial submission times are assumed to be independent. After a submission, the probability of another submission in a short time span increases for some students (self-excitation) and might decrease for others.

Your Task: Write an essay detailing a clear and structured plan on how to tackle this problem (up to 2 pages). Using the provided *UserProfile* Table and *SubmissionRecord* Table (see next pages), your task is to determine two crucial aspects of a Kaggle competition participant's submission behavior: (1) Will they resubmit their work to the leaderboard? (2) If they decide to resubmit, when will this occur?

Guide for Your Essay:

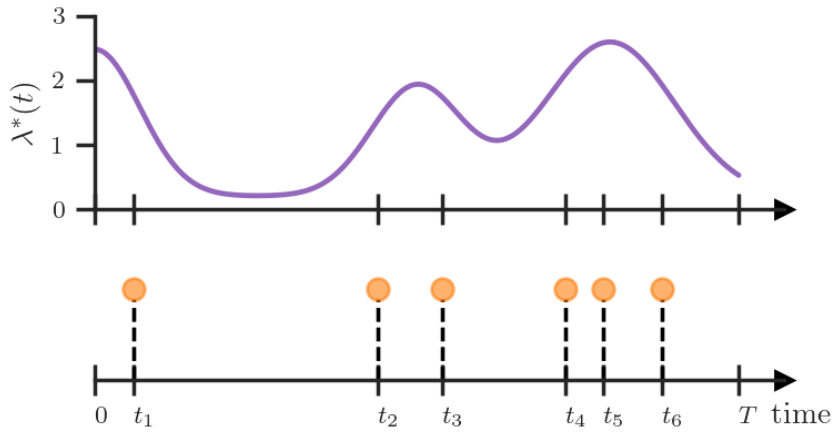
Begin your writing by deeply understanding the essence and nuances of the datasets. Contemplate the key features and patterns inherent in the data, and consider their significance in the broader context of the problem. Dive into the exploratory data analysis phase. How would you approach it to glean meaningful insights from the datasets? What visualizations or statistical tests would offer valuable perspectives on the data?

Next, detail how you would extract and engineer features pivotal for survival analysis. While crafting your approach, consider the distinct behaviors exhibited by students: some self-motivated individuals may be inclined to submit frequently as time progresses, reflecting the characteristics of the Hawkes process. On the other hand, it's essential to account for natural tendencies in the data: many participants might stop their submissions upon achieving the target score, while others may lose motivation, or even choose to ignore the assignment altogether. Recognizing these patterns, discuss how you'd use these insights in feature engineering and how they could influence the survival probabilities.

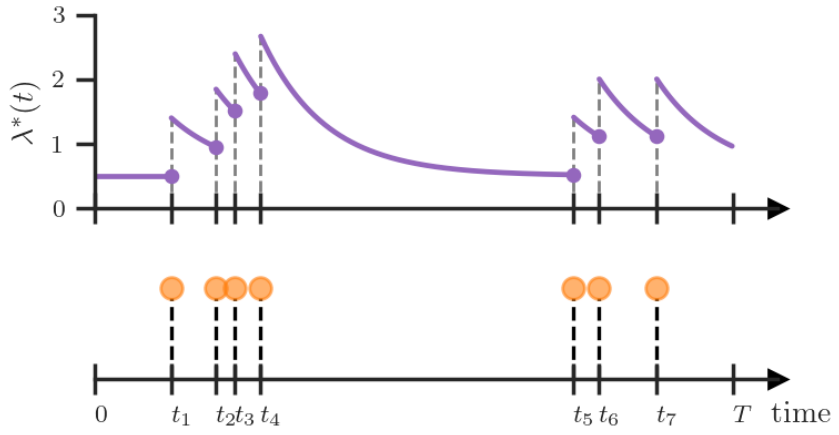
Then, delve into the intricacies of selecting and crafting a survival regression model suitable for this problem. How would you incorporate the data's inherent characteristics into your model? Given the presence of time-dependent data points and the importance of the Hawkes process, how would you adjust or modify standard survival regression techniques?

Conclude your essay by discussing strategies for model evaluation. Emphasize the importance of metrics like the c-index in this context. Moreover, reflect on the significance of various features in influencing survival probabilities and the implications of these features on the broader interpretation of the results.

Throughout your essay, prioritize a methodical and well-thought-out plan. Your writing should demonstrate deep analytical thinking, a clear understanding of the problem's intricacies, and a comprehensive blueprint detailing how you'd transition from raw data to a robust and validated model.



(a) Poisson process



(b) Hawkes process

Figure 2: A realization of a temporal point process (bottom) and the respective intensity function (top).[1]

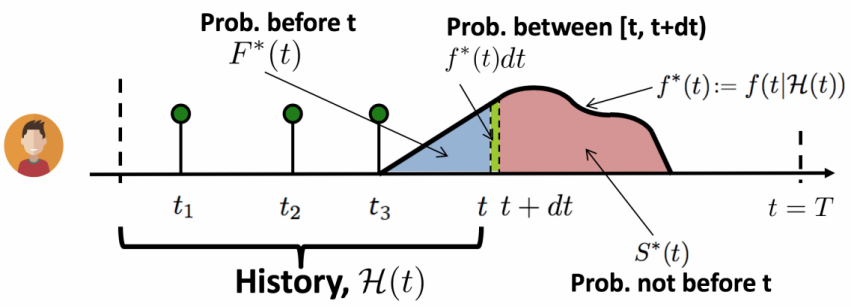


Figure 3: Visual representation of the scenario.[2] The current timestamp is denoted by t, while $\mathcal{H}(t)$ signifies the accumulated history of a participant up to that moment. The *SubmissionRecord* dataset encompasses the histories of every participant. The objective is to forecast if a participant will make another submission and, if so, the anticipated time of that submission.

[1] Figure adapted from https://shchur.github.io/blog/2020/tpp1-conditional-intensity/. See this blog post to know more about what Hawkes process is.

[2] Figure adapted from https://courses.mpi-sws.org/hcml-ws18/lectures/TPP.pdf.

Table 1: *UserProfile* table detailing the personal and academic attributes of students participating in the Kaggle competition. Each student is uniquely identified by a Unique ID. The table encompasses diverse information including the year of participation, academic status and progression (grad/undergrad and respective year), department of study, number of credits taken in the current semester, cumulative GPA, IQ, weekly study hours, and Myers-Briggs Type Indicator (MBTI) personality type. This information serves as a foundation for understanding and predicting their competition submission behaviors.

Unique ID	Status	Year	Department	# Credits	GPA	IQ	Study Hour	MBTI
1	Grad	3	Physics	12	3.8	120	25	INTJ
2	Undergrad	2	CS	15	3.5	115	20	INTP
3	Grad	4	Math	3	4.0	130	28	ENTJ
4	Undergrad	3	CS	18	2.9	110	18	ESFP
5	Grad	3	Biology	9	3.7	125	23	ISFJ
6	Grad	2	Math	10	3.6	118	22	ENFP
7	Undergrad	1	Chemistry	17	3.3	112	20	ISTJ
8	Undergrad	4	Physics	15	3.4	113	19	ESTP
9	Grad	6	Biology	0	3.9	121	27	INFJ
10	Undergrad	2	Math	18	2.8	109	17	ENTP
...	...	...	...	...	...	...	...	...

Table 2: *SubmissionRecord* table illustrating student submission behavior up to the current timestamp of *2023-11-08 18:00:00*. The preset target score is *89.0*. The competition runs from *2023-11-04* to *2023-11-12*. Notably, Student 1 made three quick submissions, exemplifying the "bursty" behavior of the Hawkes process. Student 3 had closely-timed submissions, while Student 9 showed consistent improvement. Student 6 has not submitted any results.

Unique ID	Timestamp	Accuracy Score	Beat Target
1	2023-11-08 15:05:32	85.0	No
1	2023-11-08 16:35:15	87.0	No
1	2023-11-08 17:12:10	89.5	Yes
2	2023-11-06 09:10:45	72.0	No
2	2023-11-08 16:14:25	75.2	No
3	2023-11-05 11:32:40	88.0	No
3	2023-11-05 17:01:22	89.5	Yes
3	2023-11-06 12:24:30	90.1	Yes
4	2023-11-07 14:58:21	65.3	No
5	2023-11-07 10:02:13	90.0	Yes
7	2023-11-06 08:34:23	82.0	No
7	2023-11-07 11:05:49	85.4	Yes
8	2023-11-05 19:32:53	69.0	No
9	2023-11-06 09:55:12	80.0	No
9	2023-11-07 10:25:20	81.0	No
9	2023-11-08 11:04:25	82.1	No
10	2023-11-06 15:42:31	67.0	No
...	...	...	...

Predicting submission behavior in Kaggle competitions

Kim, Minjun
20195024

In this essay, I wrote about a methodology for predicting the submission behavior of participants in Kaggle competitions. Through data analysis and feature engineering, I tried to find the factors that influence participants' submission patterns. In addition to this, I wanted to use the Hawkes Process and survival analysis to understand how participants' past behaviors and personal characteristics influence their future submission behaviors.

1 Data Distribution in *UserProfile*

Status : Graduate (Grad) and undergraduate (Undergrad) students are roughly equally distributed, encompassing students at various academic stages **(Year)**.
 # Credits : A significant number of credits are earned by most students, typically ranging between 15 to 18 credits.
GPA : A high distribution of GPAs is observed, with the majority of students maintaining a GPA of 3.0 or above.
IQ : IQ scores exhibit a wide distribution, spanning between 110 and 130.
Study Hour : Study hours are distributed between 17 and 28 hours per week.
MBTI : A roughly equal distribution is observed across different MBTI types.

2 Feature Engineering

submission_count (Number of Submissions): This feature indicates the number of submissions for each participant. It reflects the level of activity of the participants, with more active participants potentially having a higher likelihood of resubmitting.
average_event_interval (Average Time Interval Between Events): This feature calculates the average time interval between submission events for each participant. It indicates how frequently a participant submits and provides important information for the analysis of the Hawkes Process.
time_since_last_submission (Time Difference from the Last Submission to the Current Time): This feature reflects the frequency of a participant's activity up to the current moment. From the perspective of the Hawkes Process, a short time interval signifies recent activity, suggesting that the intensity of the stimulus for future activities (such as resubmissions) might be high.
time_until_deadline (Time Difference from the Last Submission to the Deadline): This feature indicates the remaining time a participant has to make additional submissions. As the deadline approaches, the submission activity of the participants may increase. This suggests that in the Hawkes Process, the temporal pressure as the deadline nears could provide a stronger stimulus for future submissions.

last_accuracy_score: This feature indicates the accuracy score of the participant's most recent submission as of today's date. Participants who have lower score might get impetus to make additional attempts of submissions in order to improve their scores. In a Hawkes Process analysis, this feature might help us understand how the score from the latest submission, up to today, could affect the likelihood and intensity of participants' future submissions.
Other features: I would like to deal with qualitative variables, for example, **Status, Department**, and **MBTI** by using one-hot encoding for the survival analysis. This will convert them to numerical form allowing the model to use them effectively.

3 Model Selection

I believe the Cox Proportional Hazards Model is well-suited for this analysis. The 'survival time' should be defined as the interval from the current moment, 18:00 on the 8th of November, until the end of the competition. This model will be used to determine the hazard ratio, predicting the timing of potential future submissions by participants. By integrating the features I've developed, this model aims to offer a detailed understanding of the various elements that might affect a participant's decision to make additional submissions before the competition's deadline.

I have considered the Hawkes Process in the feature engineering, if I can develop a Hawkes Process model, it would enable us to understand the impact of each submission on the likelihood of subsequent ones. Therefore, I believe that combining the Hawkes Process Model with the Cox Proportional Hazards Model could potentially improve the accuracies of the model as well as the effectiveness of survival analysis.

4 Model Evaluation

The Concordance Index (c-index) would be a vital metric in evaluating my model, particularly the Cox Proportional Hazards Model. The c-index assesses how well the model differentiates between participants with a higher likelihood of resubmitting and those with a lower likelihood in terms of the timing of their resubmissions.

Finally, I plan to analyze the importance of the features by identifying which ones most significantly impact the survival probabilities, which in this context refer to the probabilities of resubmission.