UGBA88 Final Exam

May 15, 2020

1 UGBA 88 Data & Decisions: Final Exam

Haas School of Business, UC Berkeley, Spring 2020

The final exam is available at ugba88.haas.berkeley.edu > shared > Exam. Copy the Exam folder to your my-work folder and open UGBA88 Final Exam.ipynb. Address each data retrieval and analysis section prompt with python code. Answer each discussion section prompt with a few thoughtful sentences. If you need any instruction clarifications or assistance with technical system issues, then contact the professor at rhuntsinger@berkeley.edu. Submit a link to your solution at bCourses > Assignments > Final Exam before 10:00pm PDT, Friday, May 15, 2020.

You are permitted to use your notes, resources on the internet, and other reference materials, but do not discuss the content of this exam with anyone until after 10:00pm PDT, Friday, May 15, 2020.

BERKELEY HONOR CODE: "As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others."

1.1 Setup

[473]:
```
# Import some useful functions
from numpy import *
from numpy.random import *
from datascience import *

# Import more useful functions for linear regression
from statsmodels.formula.api import *

# Customize look of graphics
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
%matplotlib inline

# Force display of all values
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Handle some obnoxious warning messages
import warnings
warnings.filterwarnings("ignore")
```

1.2 Hospital Planning for Needed Medicine

A city hospital treats patients who have certain symptoms using a certain medicine. In the past, it treated young patients (<18 years old), mid patients (18 to 60 years old), and senior patients (>60 years old) who had the symptoms, and kept a record of how many doses of the medicine were required for each.

The city government and hospital have developed a predictive model that estimates that next month the hospital will take in 50 to 100 young patients, 100 to 200 mid patients, and 200 to 400 senior patients, all who will have the symptoms.

1.2.1 Retrieve Data

You have a record of the hospital's past treatments available in file 'treatment_history.csv'.

Retrieve the record.

[495]:
```
data = Table.read_table('treatment_history.csv')
data
```

[495]:
```
patient_id | age   | rx
1          | young | 3
2          | young | 4
3          | young | 5
```

```
4           | young | 4
5           | young | 3
6           | young | 4
7           | young | 5
8           | young | 6
9           | young | 0
10          | young | 1
… (173 rows omitted)
```

1.2.2 Analysis

Simulate the hospital's next month activity to determine the following:

- Assuming that the hospital stocks 4000 doses of medicine, estimate the probability that 100% of patients taken in will get all the needed medicine.
- Estimate the minimum stock of medicine required to ensure that 100% of patients taken in will get all the needed medicine.

Hint: Account for the treatment record (how many doses each type of patient will need, on average) and the predictive model results (how many of each type of patient will be taken in) in your simulation.

Hint: Simulate 10000 trial scenarios for each stock decision.

Hint: You can calculate the number of new patients like this: young: `int(50 + random() * 50)` | mid: `int(100 + random() * 100)` | senior: `int(200 + random() * 200)`

Hint: You may need to estimate stock requirement through a combination of simulation and trial-and-error.

[497]:
```
young_mean = mean(data.where('age', 'young').column('rx'))
mid_mean = mean(data.where('age', 'mid').column('rx'))
senior_mean = mean(data.where('age', 'senior').column('rx'))

num_young = int(50 + random() * 50)
num_mid = int(100 + random() * 100)
num_senior = int(200 + random() * 200)

results = make_array()

for i in arange(10000):
    num_young = int(50 + random() * 50)
    num_mid = int(100 + random() * 100)
    num_senior = int(200 + random() * 200)
    total = (young_mean * num_young) + (mid_mean * num_mid) + (senior_mean *␣
 ↪num_senior)
    results = append(results, total-4000)
```

```
results_table = Table().with_columns('results', results).where('results', are.
 ↪below_or_equal_to(0))
prob_4000 = results_table.num_rows / 10000
prob_4000

max(results)

new_results = make_array()

for i in arange(10000):
    num_young = int(50 + random() * 50)
    num_mid = int(100 + random() * 100)
    num_senior = int(200 + random() * 200)
    total = (young_mean * num_young) + (mid_mean + num_mid) + (senior_mean *␣
 ↪num_senior)
    new_results = append(new_results, total - 5564)

new_results_table = Table().with_columns('new_results', new_results).
 ↪where('new_results', are.below_or_equal_to(0))
prob_5564 = new_results_table.num_rows / 10000
answer = 5564
answer
```

[497]: 0.4034

[497]: 1562.9012021857925

[497]: 5564

1.2.3 Discussion

Assume that you are Vice President of Operations, responsible for ensuring that the hospital has stocked the medicine it needs and for minimizing operating costs. How will you decide how much medicine to stock?

I will determine the amount of medicine to stock by estimating the range of medicine demand projected for the next month, then running 10000 simulations and finding the largest projected demand out of those 10000 simulations. I will stock enough medicine to meet the largest projected demand obtained through the simulations (5564), which ensures that I will almost always have enough medicine for the next month.

1.3 Certification Examination Tested by a Small Beta Group

A networking equipment company certifies people as competent to install its equipment. The company's CertDev team writes a certification exam to do this. Each question on the exam is meant to discriminate highly qualified candidates from poorly qualified candidates. Before making

the exam available to real candidates, the CertDev team asks a small group of "beta" candidates to take the exam, where each "beta" candidate is known to be either highly qualified or poorly qualified. The CertDev team then reviews the results, and is suspicious that one specific question on the exam is not a good discriminator. It would require considerable effort and expense to remove the question from the exam because the exam development process would have to be repeated to generate a replacement question.

1.3.1 Retrieve Data

The highly qualified candidates (HQC's) received these scores for the suspicious question: 1, 1, 1, 1, 0, 0, 1, 1, 1, 1

[479]:
```
high = make_array(1, 1, 1, 1, 0, 0, 1, 1, 1, 1)
mean_high = mean(high)
mean_high
std_high = std(high)
std_high
size_high = len(high)
size_high
```

[479]: 0.8

[479]: 0.4

[479]: 10

The poorly qualified candidates (PQC's) received these scores for the suspicious question: 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1

[480]:
```
poor = make_array(0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1)
mean_poor = mean(poor)
mean_poor
std_poor = std(poor)
std_poor
size_poor = len(poor)
size_poor
```

[480]: 0.45454545454545453

[480]: 0.49792959773196915

[480]: 11

1.3.2 Analysis

Use a 2-sample t hypothesis test to determine whether the suspicious question is probably a good way to distinguish highly qualified candidates from poorly qualified candidates, probably not a

good way, or cannot tell. Show the p-value to justify your conclusion.

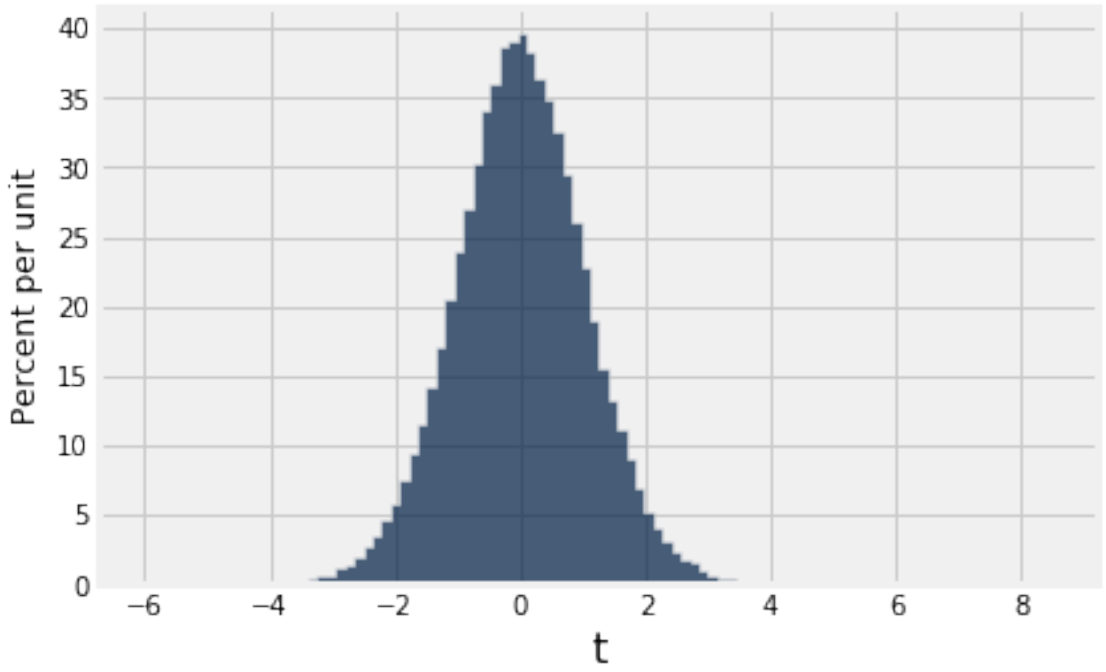
[502]:
```
mean_diff_hypo = 0
t = ((mean_high-mean_poor)-mean_diff_hypo)/sqrt((std_high**2/
 ↪size_high)+(std_poor**2/size_poor))
df = ((std_high**2/size_high)+(std_poor**2/size_poor))**2/((1/
 ↪(size_high-1))*(std_high**2/size_high)**2+(1/(size_poor-1))*(std_poor**2/
 ↪size_poor)**2)
t
values = standard_t(df, 100000)
data = Table().with_columns('t', values)
data.hist(bins=100)

p_value = data.where("t", are.above_or_equal_to(t)).num_rows / data.num_rows
sig_level = 0.05
p_value
"This question is probably a good way to distinguish highly qualified␣
 ↪candidates from poorly qualified candidates, as the p_value calculated is␣
 ↪smaller than the significance level of 0.05."
```

[502]:
```
1.7596981962070228
```

[502]:
```
0.04752
```

[502]:
```
'This question is probably a good way to distinguish highly qualified candidates
from poorly qualified candidates, as the p_value calculated is smaller than the
significance level of 0.05.'
```



1.3.3 Discussion

Assume that you are the Director of Certification Development. How will you decide whether to replace the suspicious question on the certification exam?

I will decide whether or not to replace the suspicious question on the certification exam by running a 2-sample t hypothesis test under the assumption that the mean scores between HQC's and PQC's is 0, with variation being due to chance. As shown above, the test return a p-value of 0.04702, which is smaller than the significance level of 0.05 that I chose. Therefore, I should reject the assumption that the mean scores between the two are the same, meaning that this question is probably a good way to distinguish between HQC's and PQC's.

1.4 Planning for Manufacturing Generator Replacement

A manufacturing company uses a special generator, which operates at an efficiency level that degrades over time. Maintenance is applied to the generator every 21 days, which improves the efficiency level somewhat. Immediately following maintenance, however, the efficiency level continues to degrade.

1.4.1 Retrieve Data

You have a record of the generator's efficiency level for the past 63 days available in file 'machine.csv'.

Retrieve and visualize the record as a lineplot.

[482]:
```
data = Table.read_table('machine.csv')
data
```

[482]:
```
day  | efficiency
1    | 0.927388
2    | 0.884824
3    | 0.870046
4    | 0.853197
5    | 0.843948
6    | 0.834637
7    | 0.825944
8    | 0.815438
9    | 0.811891
10   | 0.800349
… (74 rows omitted)
```

1.4.2 Analysis

Construct a simple linear regression model to predict efficiency, based on the record for days 1 through 21. The model R^2 should exceed 99%. Visualize the predicted efficiency overlayed on the record for days 1 through 21 as a lineplot.

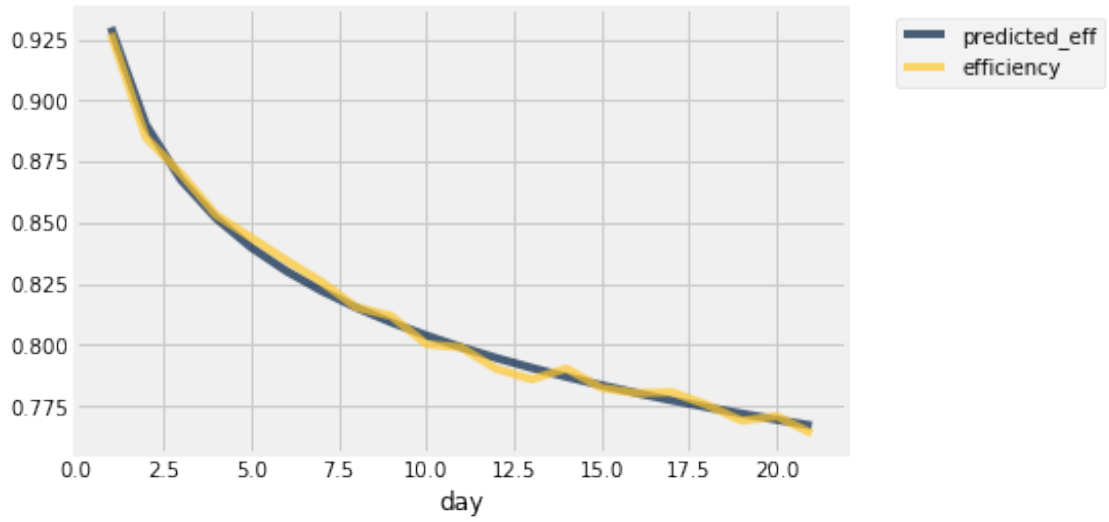
Note that the data pattern is not linear, but it may be logarithmic.

[483]:
```
data = data.where('day', are.below(22))
data = data.with_columns('log_day', log(data.column('day')), 'log_eff',␣
 ↪log(data.column('efficiency')))

model = ols('log_eff ~ log_day', data).fit()
model.rsquared

data = data.with_column('predicted_eff', exp(model.predict(data)))
data.select('day', 'predicted_eff', 'efficiency').plot('day')
```

[483]:
```
0.9945136053665098
```



Construct a simple linear regression model to expose the trend, based on the entire record. The model R^2 should exceed 58%. Visualize the trend overlayed on the entire record as a lineplot.

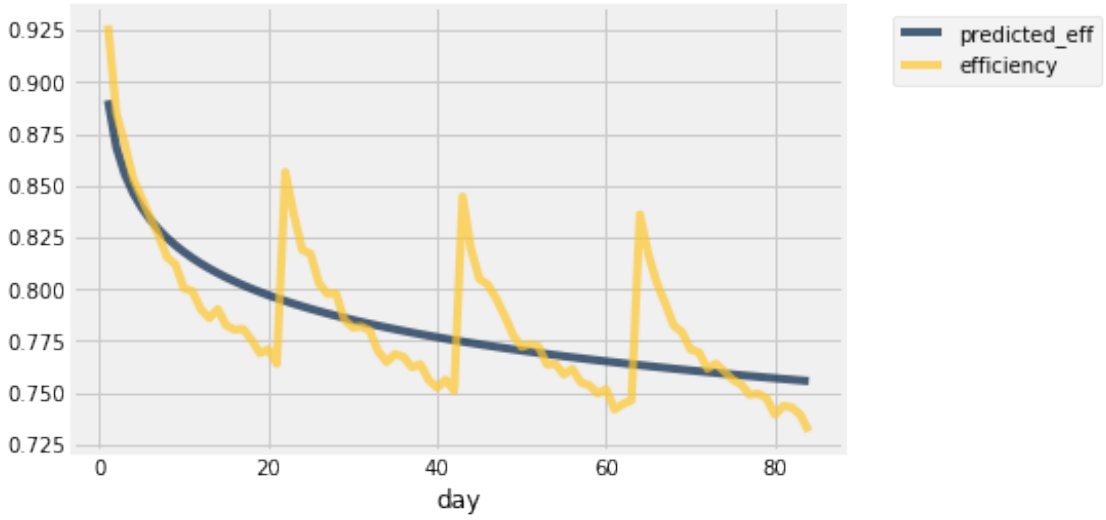
Note that the trend is not linear, but it may logarithmic.

[484]:
```
data = Table.read_table('machine.csv')
data = data.with_columns('log_day', log(data.column('day')), 'log_eff',␣
 ↪log(data.column('efficiency')))

model = ols('log_eff ~ log_day', data).fit()
model.rsquared
```

```
data = data.with_column('predicted_eff', exp(model.predict(data)))
data.select('day', 'predicted_eff', 'efficiency').plot('day')
```

[484]: 0.5779705700644668



Construct a multiple linear regression model to predict efficiency, based on the entire record. The model should account for the recurring 21-day pattern and the trend. The model R^2 should exceed 99%. Visualize the predicted efficiency overlayed on the entire as a lineplot.

Hint: To convert day to day-of-period, you can use `mod(day-1, 21) + 1`

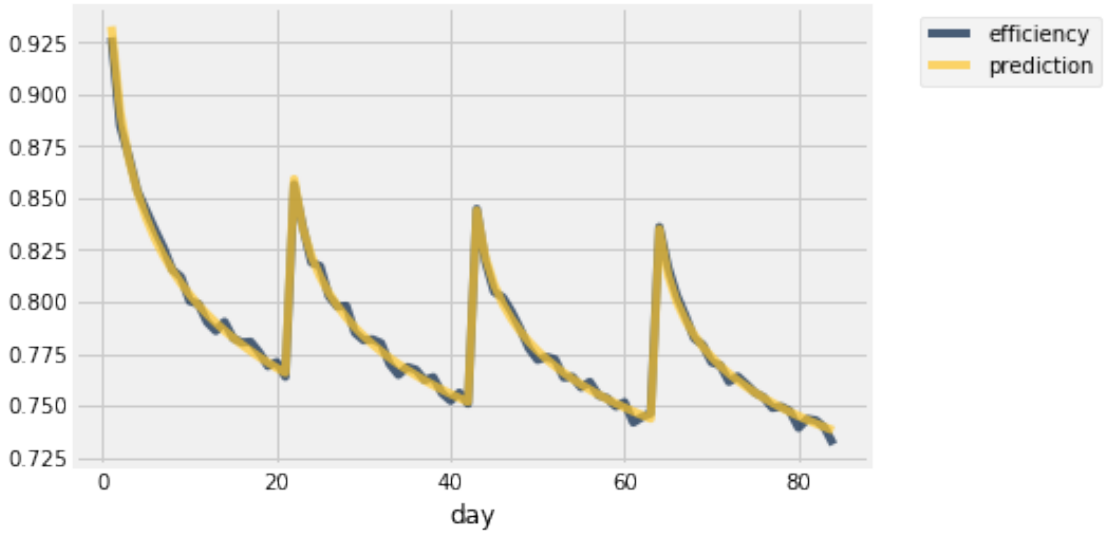
[485]:
```
data = Table.read_table('machine.csv')
data = data.with_column('day_num', log(mod(data.column('day')-1, 21)+ 1))
data = data.with_columns('log_day', log(data.column('day')), 'log_eff',␣
 ↪log(data.column('efficiency')))
data
model = ols('log_eff ~ log_day + day_num', data).fit()
model.rsquared
data = data.with_column('prediction', exp(model.predict(data)))
data = data.select('day', 'efficiency', 'prediction')
data.plot('day')
```

[485]:

day	efficiency	day_num	log_day	log_eff
1	0.927388	0	0	-0.0753836
2	0.884824	0.693147	0.693147	-0.122367
3	0.870046	1.09861	1.09861	-0.13921
4	0.853197	1.38629	1.38629	-0.158764
5	0.843948	1.60944	1.60944	-0.169664
6	0.834637	1.79176	1.79176	-0.180758

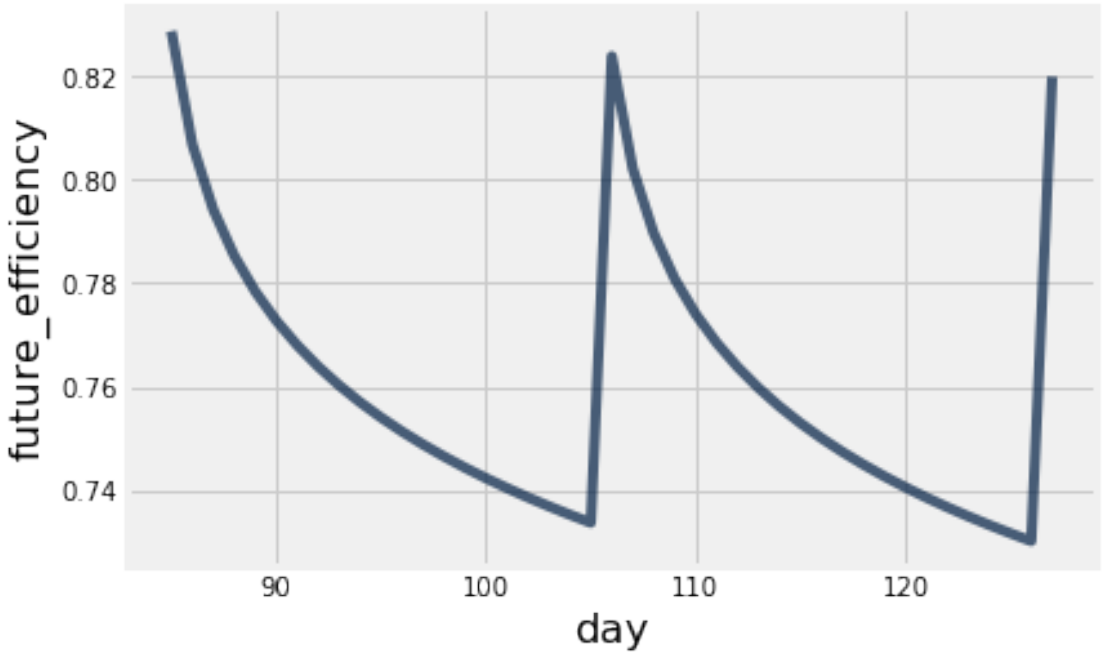
```
7    | 0.825944   | 1.94591  | 1.94591  | -0.191229
8    | 0.815438   | 2.07944  | 2.07944  | -0.20403
9    | 0.811891   | 2.19722  | 2.19722  | -0.208389
10   | 0.800349   | 2.30259  | 2.30259  | -0.222708
… (74 rows omitted)
```

[485]: 0.9932659720578985



Use the multiple linear regression model to predict efficiency for the next two 21-day periods (days 85 through 127). Visualize the predicted efficiency as a lineplot.

[486]:
```
prediction = model.predict(Table().with_columns('log_day', log(arange(1, 128,␣
 ↪1)), 'day_num', log(mod(arange(1, 128 ,1)-1, 21)+ 1)))
new_data = Table().with_columns('day', arange(1, 128, 1), 'future_efficiency',␣
 ↪exp(prediction))
new_data.where('day', are.above(84)).plot('day')
```



Use the multiple linear regression model to predict which day the generator operation will fall below a 0.73 efficiency level.

[499]:
```
prediction = model.predict(Table().with_columns('log_day', log(arange(1, 200,␣
↪1)), 'day_num', log(mod(arange(1, 200 ,1)-1, 21)+ 1)))
new_data = Table().with_columns('day', arange(1, 200, 1), 'future_efficiency',␣
↪exp(prediction))
new_data.where('future_efficiency', are.below(0.73)).column('day').item(0)
```

[499]: 146

1.4.3 Discussion

Assume you are the Vice President of Manufacturing, responsible for generator operating performance. How will you decide when to replace the generator?

I will replace the generator when its performance falls below a pre-established acceptable threshold, which in this case is 0.73. Therefore, I should replace the generator roughly every 146 days. This threshold would be calculated in advance to maximize output/profit; perhaps by analyzing the cost of maintenance against the benefits of the increased efficiency.