

DAEN 500- DL2 – Data Analytics Fundamentals

Fall 2020 Final Examination Exercise Package



FINAL EXAM PROBLEMS COMPLETE ALL & INSERT ANSWERS BELOW QUESTIONS

Problem 1: Python Programming Problem (15 Points Total)

- Design and implement a Python program that is based on the following requirements: a) program will *find all numbers -- within a specified range -- which are divisible by 7 but are not a multiple of 5*; and b) demonstrate the program works by running the program for the range: numbers between 2000 and 3200.
- **INSERT** (cut&paste) your Python code in space below and *then insert a screen shot in space below, showing your successful run and output.*

NOTE of alternative for help: To help test your code, you also may use a Python “programming window” found in the. **Zybooks Section 35 Additional Material.**

1.

The code which takes a range as an input and returns numbers which are divisible by 7 but not a multiple of 5 is shown below:

```
usermin = int(input('Enter minimum number in range: '))
usermax = int(input('Enter maximum number in range: '))
answersfromrange = []
```

```
i = 0
for i in range(usermin, (usermax + 1)):
    if i % 5 != 0:
        answersfromrange.append(i)
for i in answersfromrange:
    if i % 7 == 0:
        answersfromrange.remove(i)

print('The numbers in the range are:', i)
```

Figure 1 showing successful run in Zybooks on next page.

Figure 1

35.1 zyBooks built-in programming window

Instructor note:

Problem 1: Python Programming Problem (15 Points Total)

- Design and implement a Python program that is based on the following requirements: a) program will find all numbers which are divisible by 7 but are not a multiple of 5; and b) numbers between 2000 and 3200.
- Enter the code in the Programming window below, and "Run" the code after Pre-entering input for the program.

zyDE 35.1.1: Programming window.

[Load default template...](#)

```
1 usermin = int(input('Enter minimum number in range: '))
2 usermax = int(input('Enter maximum number in range: '))
3 answersfromrange = []
4
5 i = 0
6 for i in range(usermin, (usermax + 1)):
7     if i % 5 != 0:
8         answersfromrange.append(i)
9     for i in answersfromrange:
10         if i % 7 == 0:
11             answersfromrange.remove(i)
12
13     print('The numbers in the range are:', i)
14
```

2000
3200

Run

Enter minimum number in range: Enter
maximum number in range: The numbers in the
range are: 2002
The numbers in the range are: 2009
The numbers in the range are: 2016
The numbers in the range are: 2023
The numbers in the range are: 2037
The numbers in the range are: 2044
The numbers in the range are: 2051
The numbers in the range are: 2058
The numbers in the range are: 2072
The numbers in the range are: 2079

[Feedback?](#)



Problem 2: Python Programming Problem (15 Points Total)

- Design and implement a Python program that is based on the following requirements:
 - a) define a class which has at least two methods
 - Method 1 – getString: to get a string from console input; and,
 - Method 2 - printString: to print the string in upper case.
 - b) *demonstrate code works using three different test input strings*
- **INSERT** code below and **INSERT** a screen shot of the program and successfully run output that *includes test input for input strings (test strings must include (a) all upper case, (b) all lower case, **and** (c) mix of upper and lower case).*

For this problem, I made up several different small programs using PyCharm and Python, as seen below:

1.

Program 1 takes a cat's name and the owners name and a dog's name and the owners name and outputs the pet's and owner's names capitalized.

```
class Animal:
    def __init__(self):
        self.petname = 0
        self.ownername = 0
    def petname_and_ownername(self):
        return self.petname.upper(), self.ownername.upper()
cat1 = Animal()
cat1.petname = input('Cats name:')
cat1.ownername = input('Cats owners name:')
print('Cat:\n Name of cat and Owners name: {}'.format(cat1.petname_and_ownername()))
dog1 = Animal()
dog1.petname = input('Dogs name:')
dog1.ownername = input('Dogs owners name:')
print('Dog:\n Name of dog and Owners name: {}'.format(dog1.petname_and_ownername()))
```

Figure 2-4 show the code above working with various inputs.

Figure 2

```

pythonProject11 - problem2uppersuccess.py

1 class Animal:
2     def __init__(self):
3         self.petname = ''
4         self.ownername = ''
5
6     def petname_and_ownername(self):
7         return self.petname.upper(), self.ownername.upper()
8
9
10 cat1 = Animal()
11 cat1.petname = input('Cats name:')
12 cat1.ownername = input('Cats owners name:')
13 print('Cat:\n Name of cat and Owners name: {}'.format(cat1.petname_and_ownername()))
14
15 dog1 = Animal()
16 dog1.petname = input('Dogs name:')
17 dog1.ownername = input('Dogs owners name:')
18 print('Dog:\n Name of dog and Owners name: {}'.format(dog1.petname_and_ownername()))
19

```

Run: main

```

/Users/aleahlangrell/PycharmProjects/pythonProject11/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject11/main.py
Cats name:waldo
Cats owners name:aleah
Cat:
  Name of cat and Owners name: ('WALDO', 'ALEAH')
Dogs name:bunny
Dogs owners name:dylan
Dog:
  Name of dog and Owners name: ('BUNNY', 'DYLAN')

Process finished with exit code 0

```

Figure 3

```

pythonProject11 - problem2uppersuccess.py

1 class Animal:
2     def __init__(self):
3         self.petname = ''
4         self.ownername = ''
5
6     def petname_and_ownername(self):
7         return self.petname.upper(), self.ownername.upper()
8
9
10 cat1 = Animal()
11 cat1.petname = input('Cats name:')
12 cat1.ownername = input('Cats owners name:')
13 print('Cat:\n Name of cat and Owners name: {}'.format(cat1.petname_and_ownername()))
14
15 dog1 = Animal()
16 dog1.petname = input('Dogs name:')
17 dog1.ownername = input('Dogs owners name:')
18 print('Dog:\n Name of dog and Owners name: {}'.format(dog1.petname_and_ownername()))
19

```

Run: main

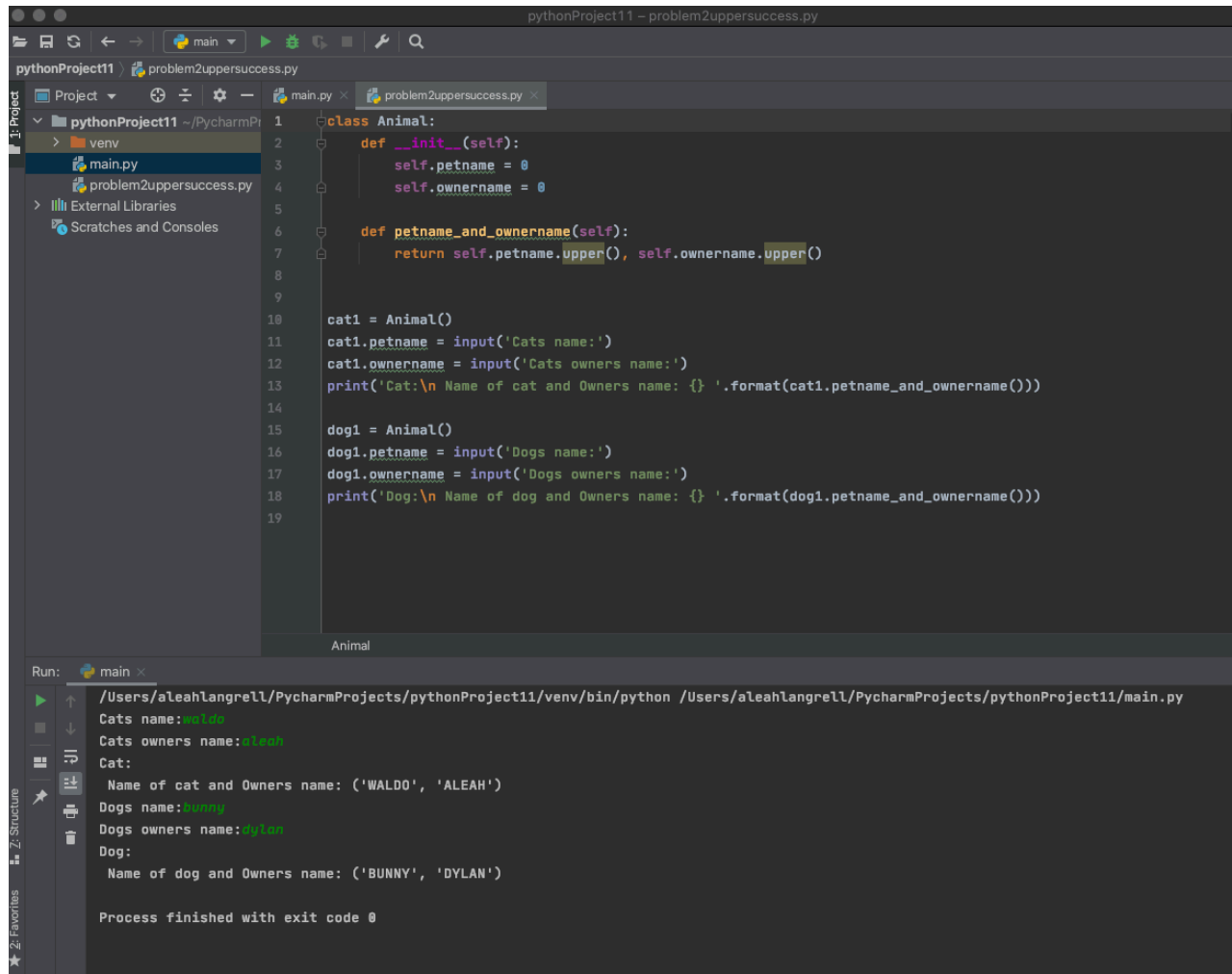
```

/Users/aleahlangrell/PycharmProjects/pythonProject11/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject11/main.py
Cats name:WALDO
Cats owners name:ALEAH
Cat:
  Name of cat and Owners name: ('WALDO', 'ALEAH')
Dogs name:BUNNY
Dogs owners name:DYLAN
Dog:
  Name of dog and Owners name: ('BUNNY', 'DYLAN')

Process finished with exit code 0

```

Figure 4



```
pythonProject11 - problem2uppersuccess.py

class Animal:
    def __init__(self):
        self.petname = 0
        self.ownername = 0

    def petname_and_ownername(self):
        return self.petname.upper(), self.ownername.upper()

cat1 = Animal()
cat1.petname = input('Cats name:')
cat1.ownername = input('Cats owners name:')
print('Cat:\n Name of cat and Owners name: {}'.format(cat1.petname_and_ownername()))

dog1 = Animal()
dog1.petname = input('Dogs name:')
dog1.ownername = input('Dogs owners name:')
print('Dog:\n Name of dog and Owners name: {}'.format(dog1.petname_and_ownername()))

Animal

Run: main x
/Users/aleahlangrell/PycharmProjects/pythonProject11/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject11/main.py
Cats name:waldo
Cats owners name:aleah
Cat:
  Name of cat and Owners name: ('WALDO', 'ALEAH')
Dogs name:bunny
Dogs owners name:dylan
Dog:
  Name of dog and Owners name: ('BUNNY', 'DYLAN')

Process finished with exit code 0
```

***Problem 2 continues on the next page.

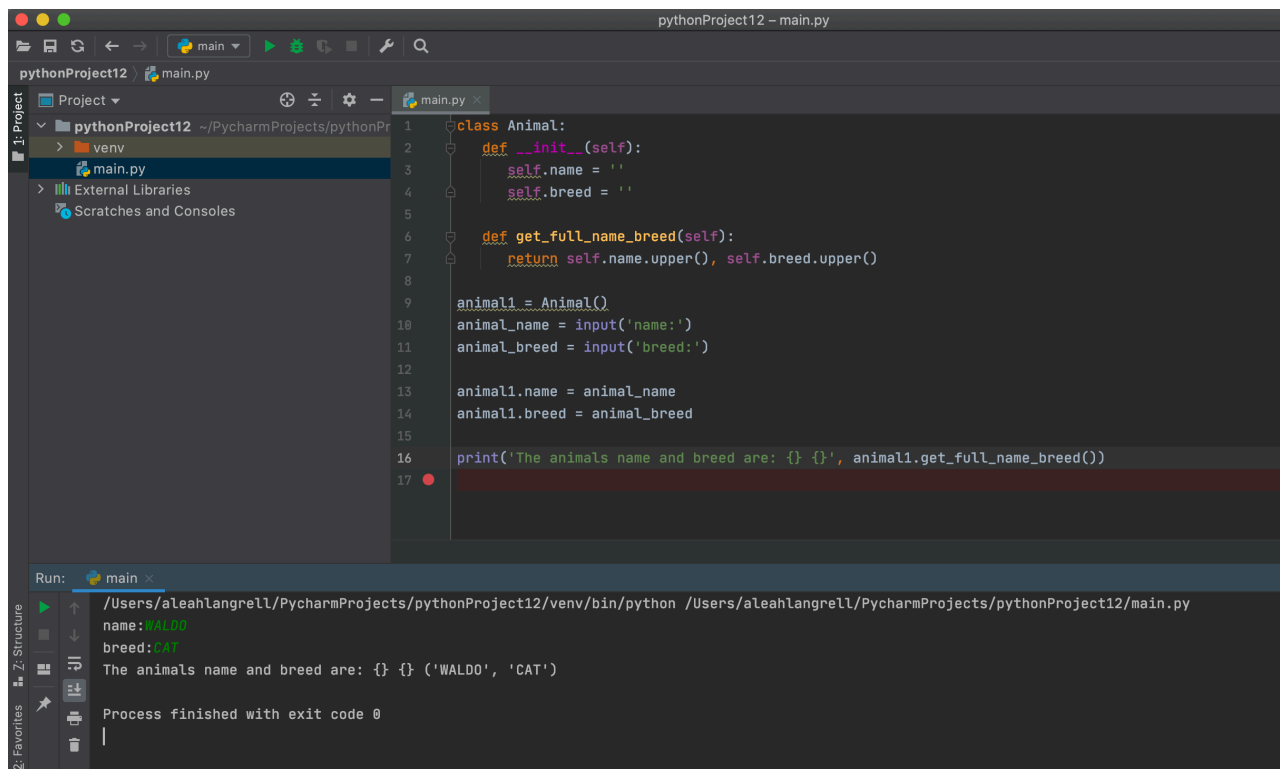
2.

Program 2 takes inputs of an animal's name and breed and outputs them in all capital letters.

```
class Animal:
    def __init__(self):
        self.name = ""
        self.breed = ""
    def get_full_name_breed(self):
        return self.name.upper(), self.breed.upper()
animal1 = Animal()
animal_name = input('name:')
animal_breed = input('breed:')
animal1.name = animal_name
animal1.breed = animal_breed
print('The animals name and breed are:', animal1.get_full_name_breed())
```

Figures 5-7 are verification the the code above works with various inputs.

Figure 5

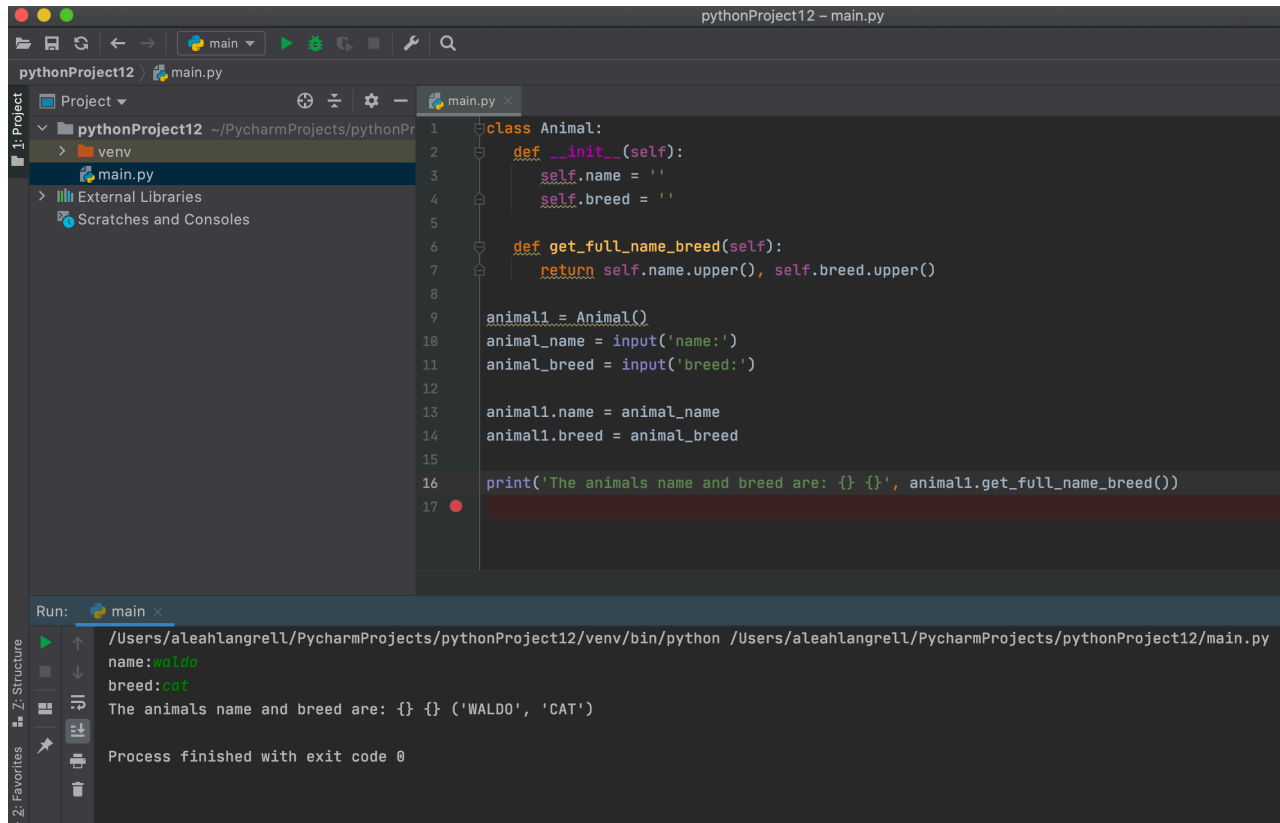


```
pythonProject12 - main.py
pythonProject12 ~/PycharmProjects/pythonProject12
venv
main.py
External Libraries
Scratches and Consoles

1 class Animal:
2     def __init__(self):
3         self.name = ""
4         self.breed = ""
5
6     def get_full_name_breed(self):
7         return self.name.upper(), self.breed.upper()
8
9 animal1 = Animal()
10 animal_name = input('name:')
11 animal_breed = input('breed:')
12
13 animal1.name = animal_name
14 animal1.breed = animal_breed
15
16 print('The animals name and breed are: {} {}'.format(*animal1.get_full_name_breed()))
17

Run: main
/Users/aleahlangrell/PycharmProjects/pythonProject12/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject12/main.py
name:WALDO
breed:CAT
The animals name and breed are: {} {} ('WALDO', 'CAT')
Process finished with exit code 0
```


Figure 6



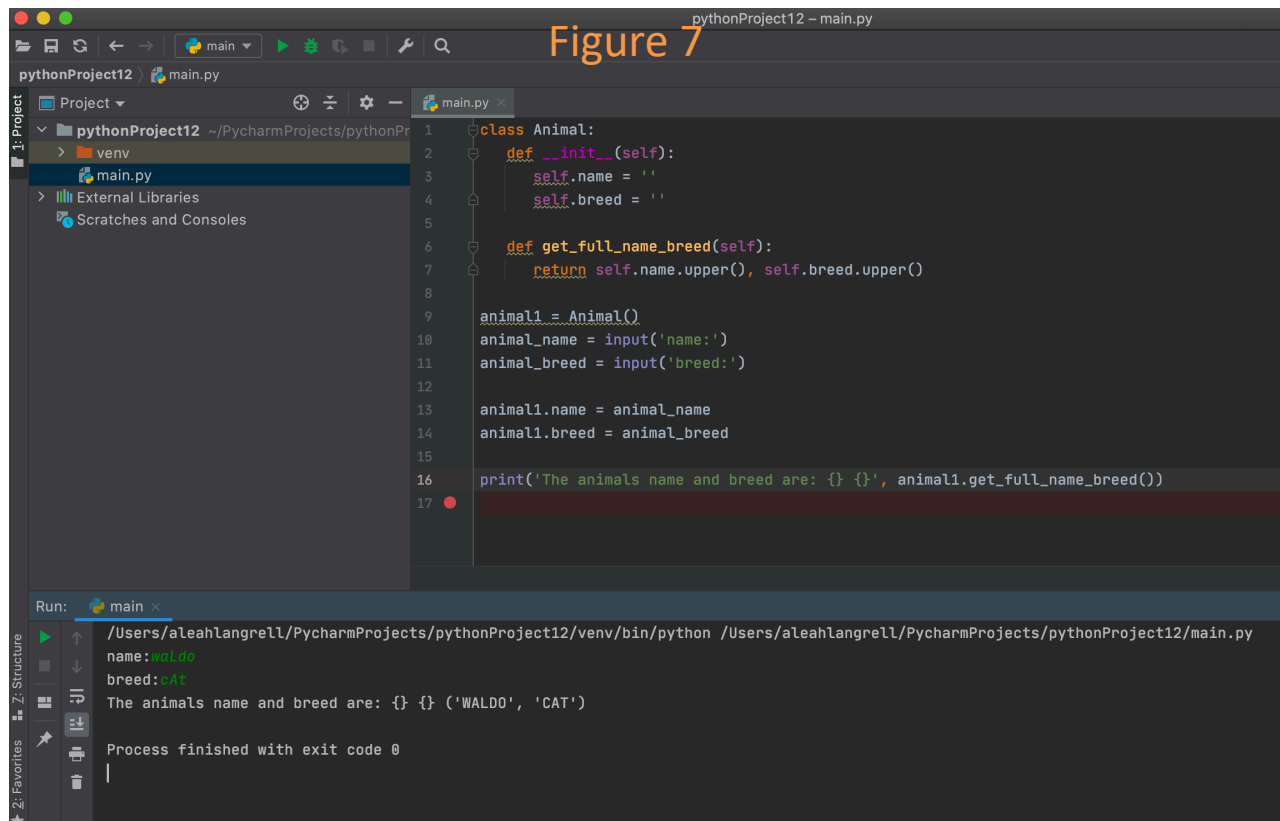
```
pythonProject12 - main.py

Project
  pythonProject12 ~/PycharmProjects/pythonProject12
    venv
    main.py
  External Libraries
  Scratches and Consoles

main.py
1 class Animal:
2     def __init__(self):
3         self.name = ''
4         self.breed = ''
5
6     def get_full_name_breed(self):
7         return self.name.upper(), self.breed.upper()
8
9 animal1 = Animal()
10 animal_name = input('name:')
11 animal_breed = input('breed:')
12
13 animal1.name = animal_name
14 animal1.breed = animal_breed
15
16 print('The animals name and breed are: {} {}'.format(*animal1.get_full_name_breed()))
17

Run: main
/Users/aleahlangrell/PycharmProjects/pythonProject12/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject12/main.py
name:waldo
breed:cat
The animals name and breed are: {} {} ('WALDO', 'CAT')
Process finished with exit code 0
```

Figure 7



```
pythonProject12 - main.py

Project
  pythonProject12 ~/PycharmProjects/pythonProject12
    venv
    main.py
  External Libraries
  Scratches and Consoles

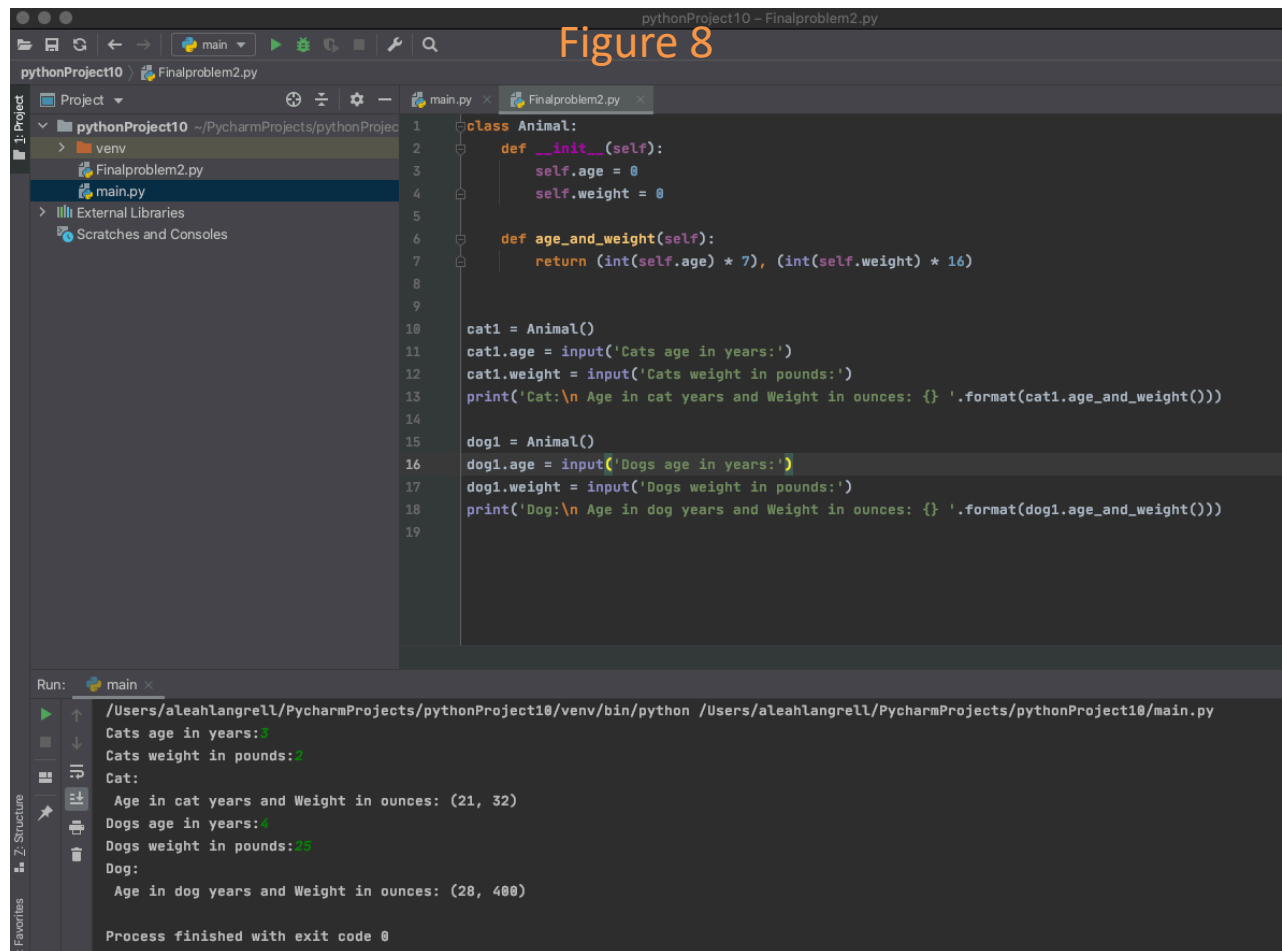
main.py
1 class Animal:
2     def __init__(self):
3         self.name = ''
4         self.breed = ''
5
6     def get_full_name_breed(self):
7         return self.name.upper(), self.breed.upper()
8
9 animal1 = Animal()
10 animal_name = input('name:')
11 animal_breed = input('breed:')
12
13 animal1.name = animal_name
14 animal1.breed = animal_breed
15
16 print('The animals name and breed are: {} {}'.format(*animal1.get_full_name_breed()))
17

Run: main
/Users/aleahlangrell/PycharmProjects/pythonProject12/venv/bin/python /Users/aleahlangrell/PycharmProjects/pythonProject12/main.py
name:waldo
breed:cat
The animals name and breed are: {} {} ('WALDO', 'CAT')
Process finished with exit code 0
```

3.

Program 3 was written to take an animal's age in years and weight in pounds and convert the animal's age in human years and the animal's weight in ounces. (Seen in Figure 8)

```
class Animal:
    def __init__(self):
        self.age = 0
        self.weight = 0
    def age_and_weight(self):
        return (int(self.age) * 7), (int(self.weight) * 16)
cat1 = Animal()
cat1.age = input('Cats age in years:')
cat1.weight = input('Cats weight in pounds:')
print('Cat:\n Age in cat years and Weight in ounces: {}'.format(cat1.age_and_weight()))
dog1 = Animal()
dog1.age = input('Dogs age in years:')
dog1.weight = input('Dogs weight in pounds:')
print('Dog:\n Age in dog years and Weight in ounces: {}'.format(dog1.age_and_weight()))
```





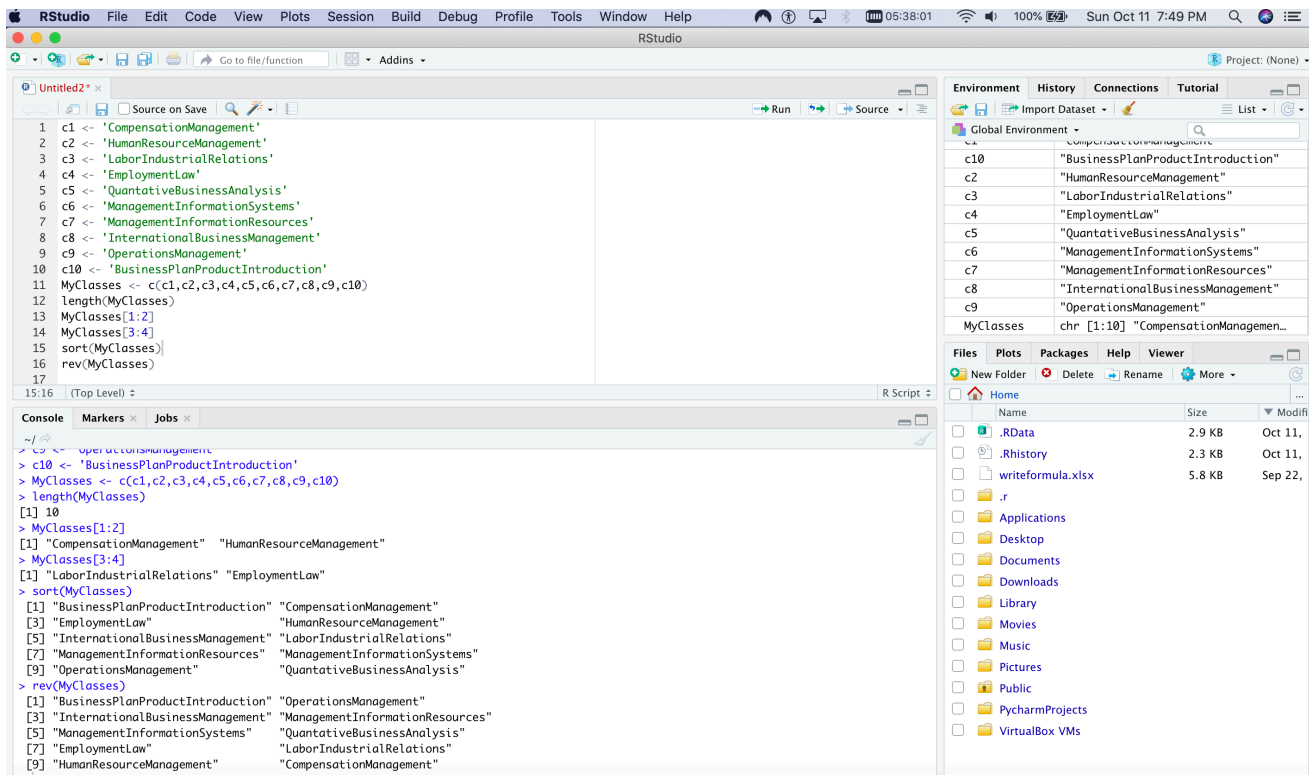
Problem 3: R Programming Problem (20 Points Total)

- **Perform the following problems using R:**
 - Create a vector of courses (e.g., MATH 101) you have taken previously. Make sure you have at least 8 courses. Name the vector myCourses
 - Get the length of the vector myCourses
 - Get the first two courses from myCourses
 - Get the 3rd and 4th courses from myCourses
 - Sort myCourses using a method
 - Sort myCourse in the reverse direction

Code for R Programming Problem:

```
>c1 <- 'CompensationManagement'
>c2 <- 'HumanResourceManagement'
>c3 <- 'LaborIndustrialRelations'
>c4 <- 'EmploymentLaw'
>c5 <- 'QuantativeBusinessAnalysis'
>c6 <- 'ManagementInformationSystems'
>c7 <- 'ManagementInformationResources'
>c8 <- 'InternationalBusinessManagement'
>c9 <- 'OperationsManagement'
>c10 <- 'BusinessPlanProductIntroduction'
>MyClasses <- c(c1,c2,c3,c4,c5,c6,c7,c8,c9,c10)
>length(MyClasses)
>MyClasses[1:2]
>MyClasses[3:4]
>sort(MyClasses)
>rev(MyClasses)
```

Figure 9





Problem 4: Principal Component Analysis (25 points)

Provide a description of the following:

- 1) What is a component – Provide a description (5 points)
- 2) Principal Component Analysis – Provide a description.(5 points)
- 3) **Provide and explain an specific example of a Principal Component Analysis(15 points)**

1.

A **component** (Z_i) is a new variable calculated by correlation values that serve as a “weighted combination of predictor variables” which are used to reduce the amount of parameters in an analysis problem (Zybooks, Ch9, 2020 & one of the LinkedInLearning videos assigned this semester). A **principal component** ($Z_i = v_{i,1}X_1 + v_{i,2}X_2$) (used in principal component analysis) is a set of correlated variables that are combined based on their weight (v_i) which comes from the magnitude of the effect of increasing or decreasing the original variables (X) on the principal component (Zybooks, Ch. 22.1, 2016)

2.

A **principal component analysis** is an analysis that can be used on large datasets that have variables with unknown correlations to reduce the number of variables to include in the formula for the analysis (Zybooks, 2016). We also use principal component analysis to combine multiple correlated variables into one component and to reduce collinearity (LinkedInLearning, 2020).

3.

The only example of a principal component analysis that zybooks provided was based on the question of whether freshman at a university would succeed in their academics given data about their test scores, GPA, clubs, and volunteering. This involved comparing the weights of the independent variables, which found test scores and GPA, and clubs and volunteering to be correlated with one another. Setting up a correlation table is a great way to visualize variable relations.

An eigenvector (v) is the vector that “corresponds to a scaling factor, lambda, called the eigenvalue [...] and the eigenvectors for a matrix will be orthogonal to each other” (Zybooks, Ch. 22.2, 2016). Next the axes with the highest variability are found using the correlation matrix to move the data.

(Zybooks, 2016) Next, the eigenvalues, which show the variance of each of the variables, were found from the eigenvector to evaluate the set of correlated variables to determine the highest value, which would be the first principal component. The amount of variables equals the eigenvalues added together, and the higher the eigenvalue the higher the variability of that variable.

The correlation matrix is found by comparing each pair of values, which results in each of the comparisons equaling a new value and when all the pairs have been compared the correlation matrix is complete.

The variables are then manipulated to have a mean and standard deviation of zero in order to be able to properly examine the variables using the same scale. Factor loadings can also be performed to identify correlation between the new principal components and the variables in the initial dataset. The factor loadings squared equals the “percent variance in the input variable contained in the principal component” (Zybooks, Ch. 22.5, 2016).



Problem 5: Multiple vs. Logistic (30 points)

- (a) **Describe:** What is difference between Multiple Regression and Logistic Regression? What circumstances might determine which to use? (10 points)
- (b) **Demonstrate:** Using any data, and any tool set you've learned about, show differences (20 points)

SUGGESTION: may be solved using RapidMiner, or other toolsets, BOTH TO ANALYZE AND TO VISUALIZE REGRESSION DIFFERENCES..

Step 1: Perform a quick search of the [UCIS public data archive](#), a well-curated site which you already have seen as part of your introductory RapidMiner training.

Step 2: Pick a dataset you find interesting, input dataset into regression tools you've chosen.

Step 3: Run the dataset (*may be a significant subset, if the dataset is very large*) first. a Multiple Regression and then a Logistic Regression, .and use visualizations to **demonstrate** the conceptual answers you provided for 5.(a).

A.)

I. A **multiple linear regression** is a function used to model the relationship between one dependent variable (y) and independent variables (x_1, x_2, \dots) to evaluate how increasing or decreasing the independent variables affects the dependent variable. The multiple linear regression model for the sample represents our prediction of the dependent variable by setting it (\hat{Y}) equal to an intercept (β_0) plus the sum of each independent variable (X_i) multiplied by its slope (β_i) plus the error term (E). The actual value of the dependent variable and our predictions of the dependent variable (which represents all the outside variables that were not accounted for, but contributed to the value of the dependent variable) are observed in the regression errors which are not related to the regression error of any of the other variables (Zybooks, 2016).

II. A **logistic regression** is a function that is more useful for data that contains a binary response dependent variable (variables with only 2 outcomes, i.e TRUE/FALSE, YES/NO, etc.) that are converted into dummy variables. Binary data forms "S" shaped graph outcomes, because as the dependent variable increases or decreases, the proportion will not go on forever (0 to 1). Like the multiple linear regression model, the logistic regression model consists of two parts: the regression function showing the probability proportion of the dependent binary variable and the logistic regression error that shows the "difference between the binary response variable and the probability from the logistic regression function" (Zybooks, Ch24.1, 2016).

III. The type of the dependent variable is one of the main considerations when deciding whether to use a logistic or multiple linear regression. If the dependent variable is an integer and consists of numerical values with few identical values, then multiple regression is better; but if the dependent variable is binary or has only two outcomes, logistic regression is better. The size of the data, number of independent variables and reason for the analysis can also help identify which model to use.

B.)

I. After performing a quick search of the UCIS website, I found a couple of datasets to work with: one consisting of dermatological data (the type of data for a logistic regression model) and one with computer machine data (the appropriate type of data for a multiple linear regression). The link to the data I chose to work with for this project is provided below.

Computer machine data: <https://archive.ics.uci.edu/ml/datasets/Computer+Hardware>

II. I chose to run both types of regressions on the computer machine data to turn in. The computer machine data compared an estimated prediction of performance and published performance in relation to the vendor of the machine, the model of the machine, the machine cycle time, the minimum and maximum memory, the cache memory, the minimum and maximum channels.

III. Using Rapidminer, I imported the data for the computer machine. Using the computer machine data to run a linear regression in RapidMiner, as seen in Figure 10, was simple and returned p values of 0 for a couple of variables including the company micro data, the minimum memory, maximum memory, cache and channel maximum. The linear regression model equation for the computer machine data using only significant p values is:

$$\text{Estimated Relative Performance} = -55.33 + 2.498 * \text{Channel max} + .74 * \text{cache} + .004 * \text{memorymax} + .019 * \text{memorymin} - 264.79 * \text{microdata (vendor)}$$

The R^2 of this process found in RapidMiner as squared_correlation is 0.835, meaning that the linear regression model predicts the correct value of the dependent variable 83.5% of the time. The correlation was found to be 0.914. Figure 11 is a chart showing the correlation showing the estimated prediction on the y axis and the published performance on the x axis with cache, one of the significant p-values, plotted and a line of best fit.

Additionally, I imported the dataset into R to analyze and graph the data to make sure that the work in RapidMiner was done correctly, as seen in Figure 12.

Figure 10

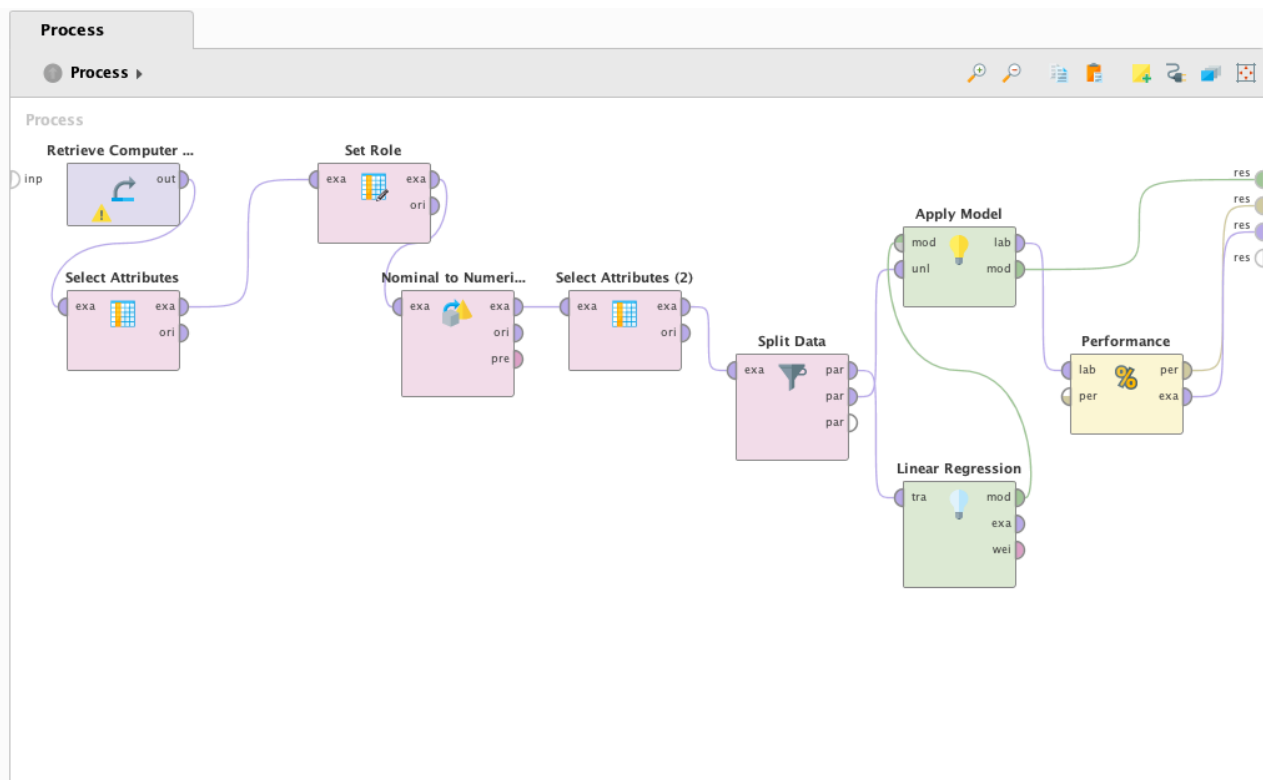
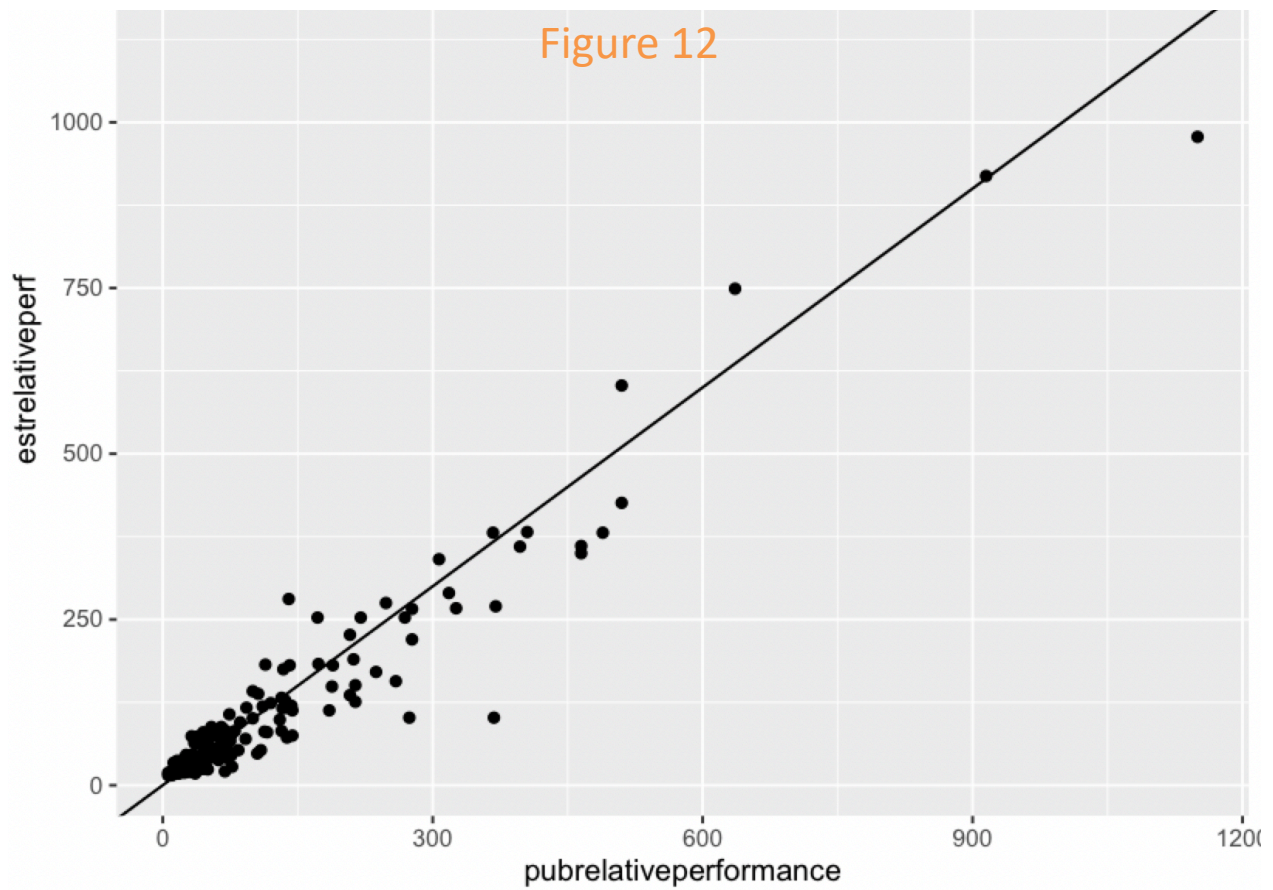
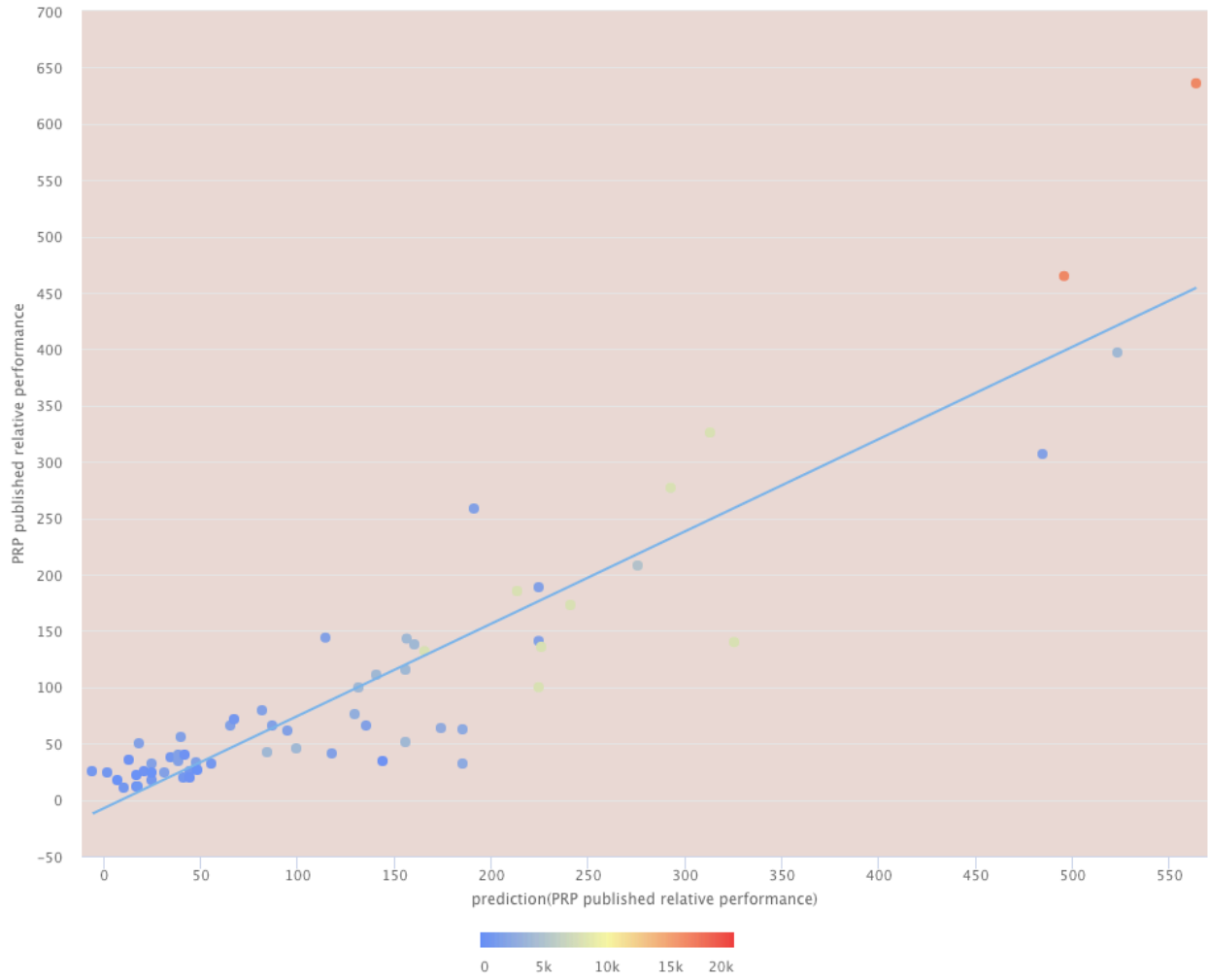


Figure 11
Published vs Prediction



Using the same dataset to run a logistic regression, was not as simple. Since the computer dataset was a better fit for linear regression, and did not have a binary dependent variable, I had to change the dependent variable, estimated relative performance, to a binomial. Figure 13 shows the process for the logistic regression in RapidMiner. When the processes were finished and ran, all of the p-values returned were 1, indicating that a logistic regression is not a good fit for this particular data. There were no visualizations available, which can possibly be attributed to running the wrong type of operator for the data imported. Since there were no visualizations for the logistic regression, Figure 15 shows the description from the logistic regression model in RapidMiner.

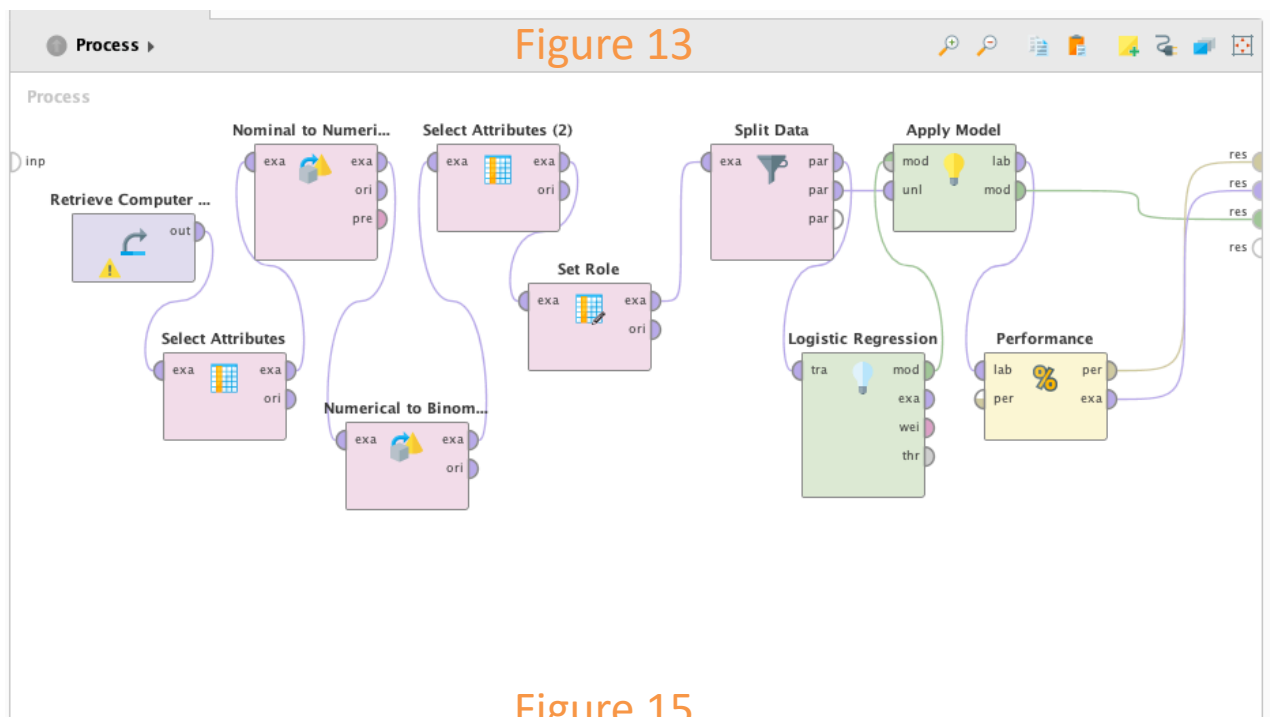


Figure 15

Logistic Regression Model

```

Model Metrics Type: BinomialGLM
Description: N/A
model id: rm-h2o-model-logistic_regression-439391
frame id: rm-h2o-frame-logistic_regression-408665
MSE: 0.0
RMSE: 0.0
R^2: NaN
AUC: 0.0
pr_auc: 0.0
logloss: 0.0
mean_per_class_error: NaN
default threshold: 0.5
Gains/Lift Table (Avg response rate: 0.00 %, avg score: 100885.54 %):
  Group  Cumulative Data Fraction  Lower Threshold  Lift  Cumulative Lift  Response Rate  Score  Cumulative Response Rate  Cumulative Score C
    1      1.000000000      1008.855402  NaN      NaN      0.000000  1008.855402      0.000000      1008.855402
null DOF: 145.0
residual DOF: 145.0
null deviance: 0.0
residual deviance: 0.0
GLM Model (summary):
  Family  Link  Regularization  Number of Predictors  Total Number of Active Predictors  Number of Iterations  Training Frame
binomial  logit      None              34              0              0      rm-h2o-frame-logistic_regression-408665
Scoring History:
  timestamp  duration  iterations  negative_log_likelihood  objective
2020-10-13 18:24:58  0.000 sec      0      0.00000  0.00000
H2O version: 3.30.0.1-rm9.7.1

```

References

Pardoe, I., Sturdivant, R., Berrier, J., Nestler, S., Watts, K., Vahid, F., & Chan, C. (2016) Data Analytics Fundamentals in *Zybooks: Chapters 22-24*. Retrieved from <https://learn.zybooks.com/zybook/GMUDAEN500BerlinFall2020>

Feldmesser, J. & Ein-Dor, P. (1987) Computer Hardware Data Set on *UCI Machine Learning Repository*. Retrieved on October 7, 2020 from: <https://archive.ics.uci.edu/ml/datasets/Computer+Hardware>