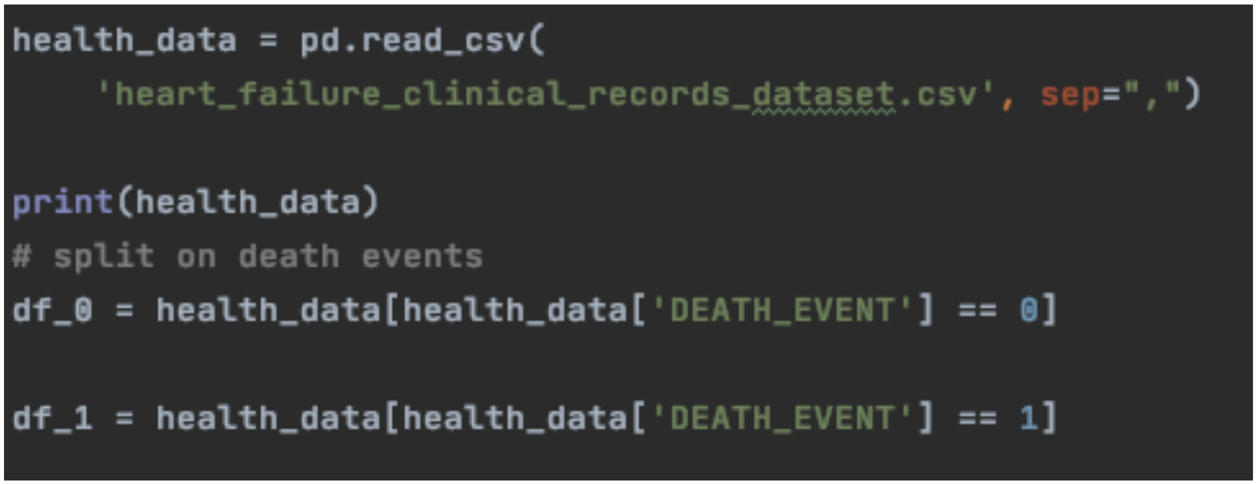
Justin Rebollo
CS 677
4/12/21

Homework #4

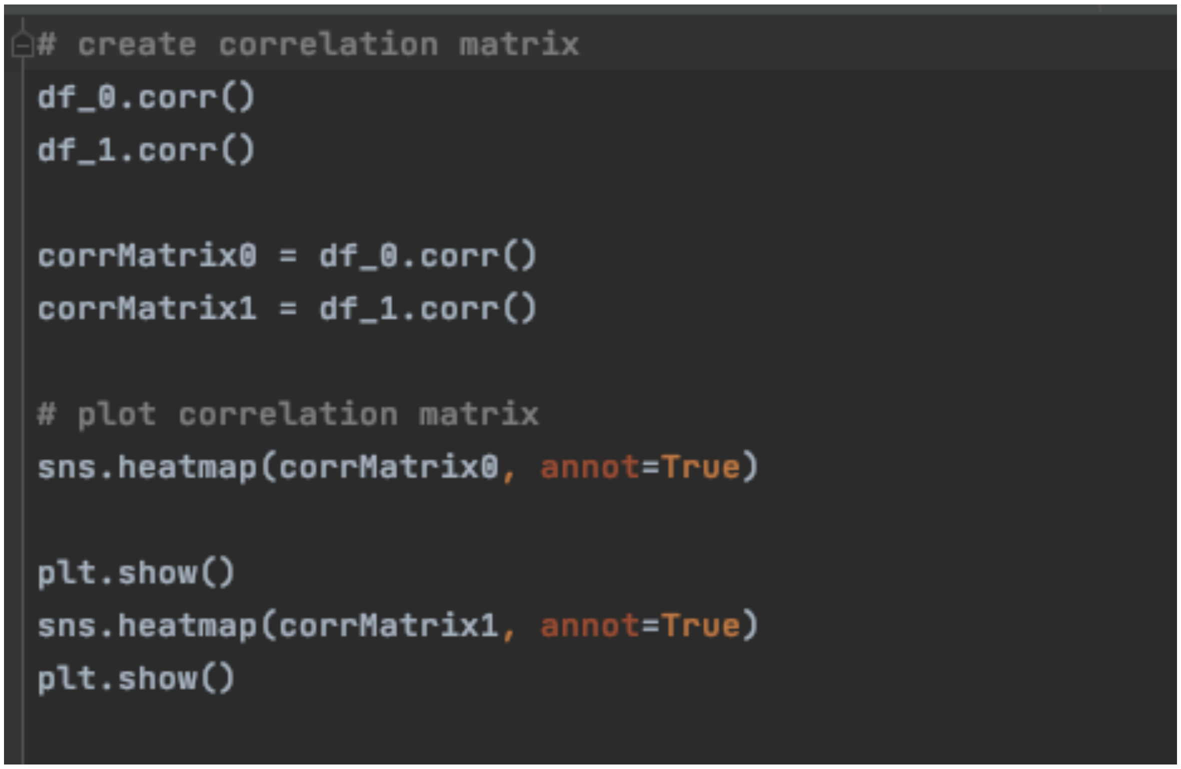
Question 1.

1. See code 1_1



```
health_data = pd.read_csv(
    'heart_failure_clinical_records_dataset.csv', sep=",")

print(health_data)
# split on death events
df_0 = health_data[health_data['DEATH_EVENT'] == 0]

df_1 = health_data[health_data['DEATH_EVENT'] == 1]
```

2. See code 1_2



```
# create correlation matrix
df_0.corr()
df_1.corr()

corrMatrix0 = df_0.corr()
corrMatrix1 = df_1.corr()

# plot correlation matrix
sns.heatmap(corrMatrix0, annot=True)

plt.show()
sns.heatmap(corrMatrix1, annot=True)
plt.show()
```

3. Questions below

- (a) which features have the highest correlation for surviving patients?

  Smoking and sex of patient have the highest correlation for surviving patients at .49

- (b) which features have the lowest correlation for surviving patients?

  Time and serum creatinine have the lowest correlation for surviving patients at .000086

- (c) which features have the highest correlation for deceased patients?

  Smoking and sex of patient have the highest correlation for surviving patients at .36

- (d) which features have the lowest correlation for deceased patients?

  Diabetes and serum sodium have the lowest correlation for surviving patients at .000086

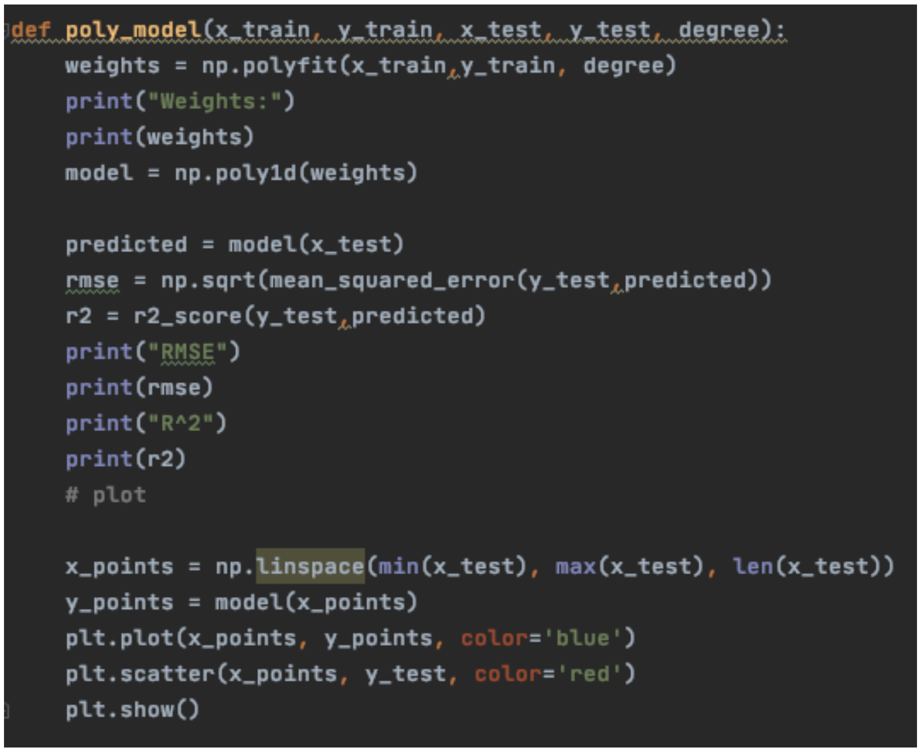
- (e) are results the same for both cases?

  The results are only the same in both cases for the features with the highest correlation. Smoking and sex of patient has the highest correlation across both surviving and deceased patients. The lowest correlation between features for both categories of patient are different.

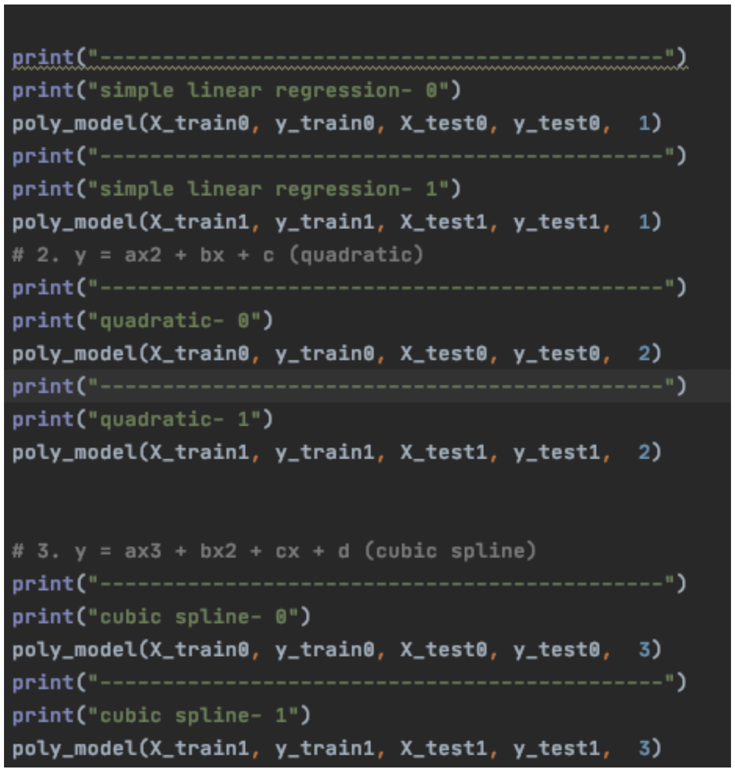
Question 2.

See code 2_1 in rebollo_hw_4

Linear and Polynomial

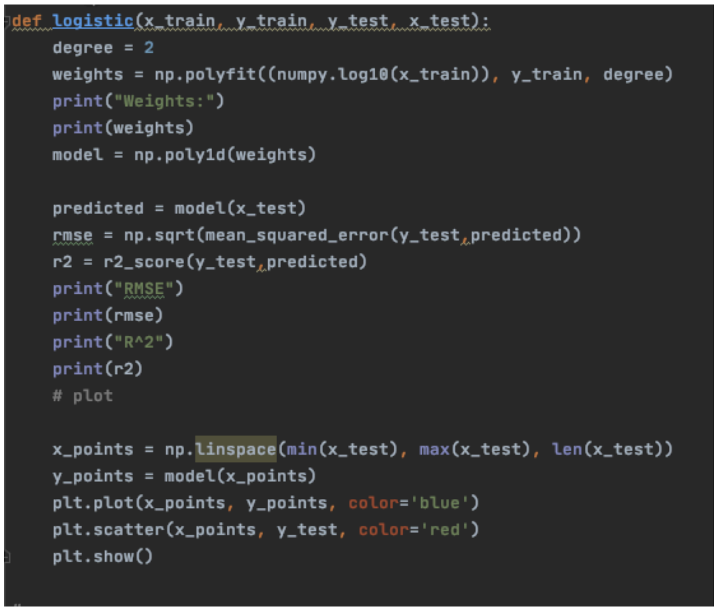


```python
def poly_model(x_train, y_train, x_test, y_test, degree):
    weights = np.polyfit(x_train,y_train, degree)
    print("Weights:")
    print(weights)
    model = np.poly1d(weights)

    predicted = model(x_test)
    rmse = np.sqrt(mean_squared_error(y_test,predicted))
    r2 = r2_score(y_test,predicted)
    print("RMSE")
    print(rmse)
    print("R^2")
    print(r2)
    # plot

    x_points = np.linspace(min(x_test), max(x_test), len(x_test))
    y_points = model(x_points)
    plt.plot(x_points, y_points, color='blue')
    plt.scatter(x_points, y_test, color='red')
    plt.show()
```

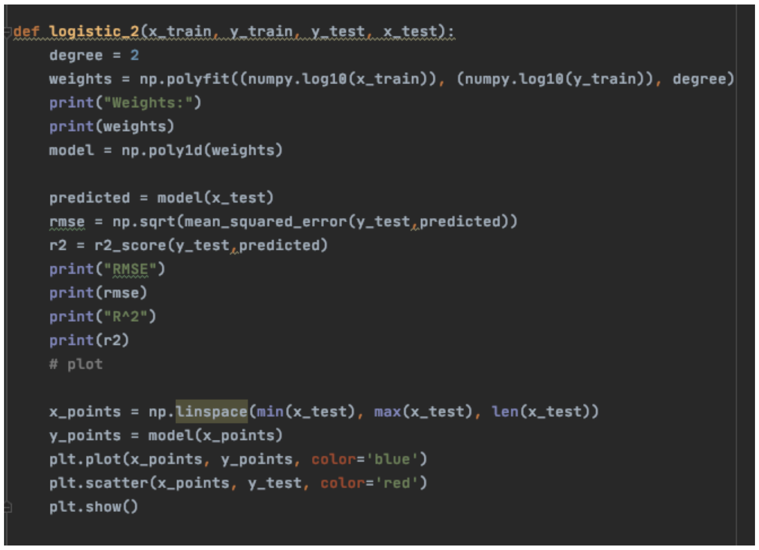


```python
print("---------------------------------------------")
print("simple linear regression- 0")
poly_model(X_train0, y_train0, X_test0, y_test0,  1)
print("---------------------------------------------")
print("simple linear regression- 1")
poly_model(X_train1, y_train1, X_test1, y_test1,  1)
# 2. y = ax2 + bx + c (quadratic)
print("---------------------------------------------")
print("quadratic- 0")
poly_model(X_train0, y_train0, X_test0, y_test0,  2)
print("---------------------------------------------")
print("quadratic- 1")
poly_model(X_train1, y_train1, X_test1, y_test1,  2)


# 3. y = ax3 + bx2 + cx + d (cubic spline)
print("---------------------------------------------")
print("cubic spline- 0")
poly_model(X_train0, y_train0, X_test0, y_test0,  3)
print("---------------------------------------------")
print("cubic spline- 1")
poly_model(X_train1, y_train1, X_test1, y_test1,  3)
```

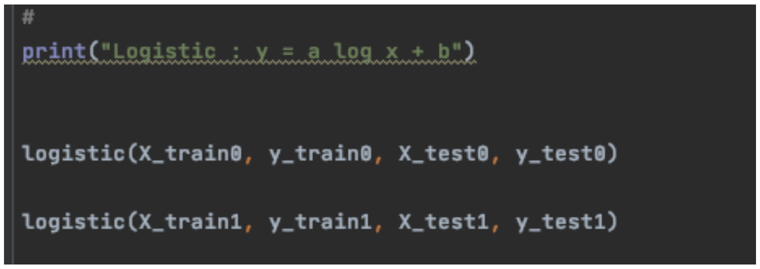
Logistic Models-



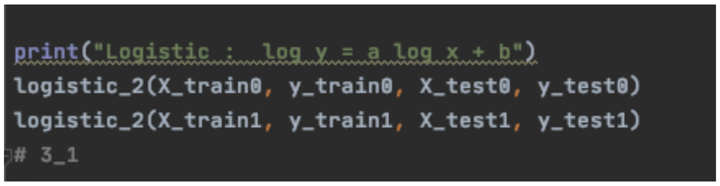
```
def logistic(x_train, y_train, y_test, x_test):
    degree = 2
    weights = np.polyfit((numpy.log10(x_train)), y_train, degree)
    print("Weights:")
    print(weights)
    model = np.poly1d(weights)

    predicted = model(x_test)
    rmse = np.sqrt(mean_squared_error(y_test,predicted))
    r2 = r2_score(y_test,predicted)
    print("RMSE")
    print(rmse)
    print("R^2")
    print(r2)
    # plot

    x_points = np.linspace(min(x_test), max(x_test), len(x_test))
    y_points = model(x_points)
    plt.plot(x_points, y_points, color='blue')
    plt.scatter(x_points, y_test, color='red')
    plt.show()
```



```
def logistic_2(x_train, y_train, y_test, x_test):
    degree = 2
    weights = np.polyfit((numpy.log10(x_train)), (numpy.log10(y_train)), degree)
    print("Weights:")
    print(weights)
    model = np.poly1d(weights)

    predicted = model(x_test)
    rmse = np.sqrt(mean_squared_error(y_test,predicted))
    r2 = r2_score(y_test,predicted)
    print("RMSE")
    print(rmse)
    print("R^2")
    print(r2)
    # plot

    x_points = np.linspace(min(x_test), max(x_test), len(x_test))
    y_points = model(x_points)
    plt.plot(x_points, y_points, color='blue')
    plt.scatter(x_points, y_test, color='red')
    plt.show()
```



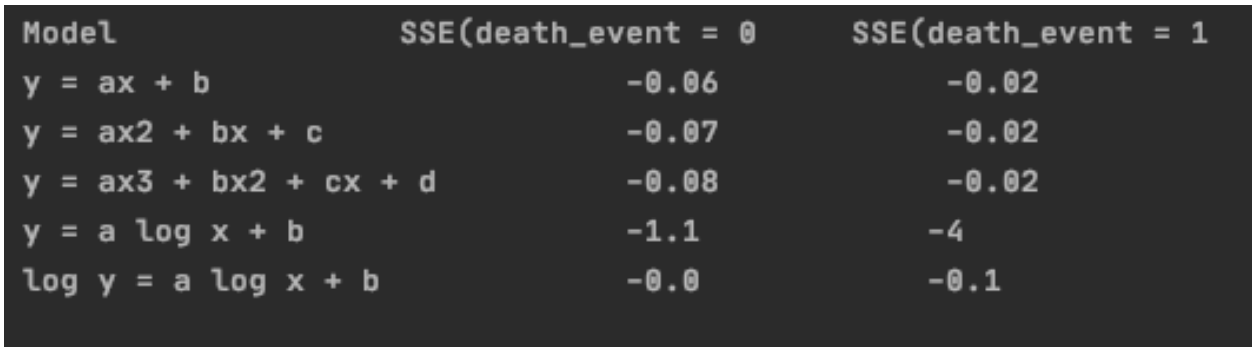
```
#
print("Logistic : y = a log x + b")


logistic(X_train0, y_train0, X_test0, y_test0)

logistic(X_train1, y_train1, X_test1, y_test1)
```



```
print("Logistic :  log y = a log x + b")
logistic_2(X_train0, y_train0, X_test0, y_test0)
logistic_2(X_train1, y_train1, X_test1, y_test1)
# 3_1
```

Question 3.

Output table for my instance



```
Model                     SSE(death_event = 0     SSE(death_event = 1
y = ax + b                            -0.06                -0.02
y = ax2 + bx + c                      -0.07                -0.02
y = ax3 + bx2 + cx + d                -0.08                -0.02
y = a log x + b                       -1.1                -4
log y = a log x + b                   -0.0                -0.1
```

1. which model was the best (smallest SSE) for surviving patients? for deceased patients?

The best SSE for surviving patients is the log y = a log x + b model and the best for the deceased is the log y = a log x + b as well.

2. which model was the worst (largest SSE) for surviving patients? for deceased patients?

The worst SSE for surviving patients is the logarithmic model and the worst for the deceased is the logarithmic model as well.