Exam 2, Portion 2 - Samuel Moreno

(5 points): If you are not familiar with this kind of scenario, but you were hired for this job in the real world, you would need to learn about it. Find two resources on the internet that are reliable and of good research quality (no forums, blogs, videos, social media, etc...)

Source 1: https://pmc.ncbi.nlm.nih.gov/articles/PMC4953449/

Key takeaways: Readmitted patientss are older on average, initial stay was longer for readmitted patient, an overwhelming majority of readmitted patients could have been avoided, readmission without conclusive therapy was the leading cause for readmission.

Source 2: https://pmc.ncbi.nlm.nih.gov/articles/PMC4953449/

Key takeaways: Around 20% of medicare beneficieares experience readmission within 30 days. Hospitals have been taken preventive measures to try to avoid readmissions being as frequent, and have reduced the percentage by about 5% over the years.

"The most common preventable factors were emergency department decision-making regarding readmission, failure to relay important information to outpatient providers, discharge of patients too soon, and lack of goals of care discussions among patients with serious illnesses. "

(5 points): Download this dataset and assess it using ISLP 3.3 and 3.4 (like how you did for homework 3).

In [1]:

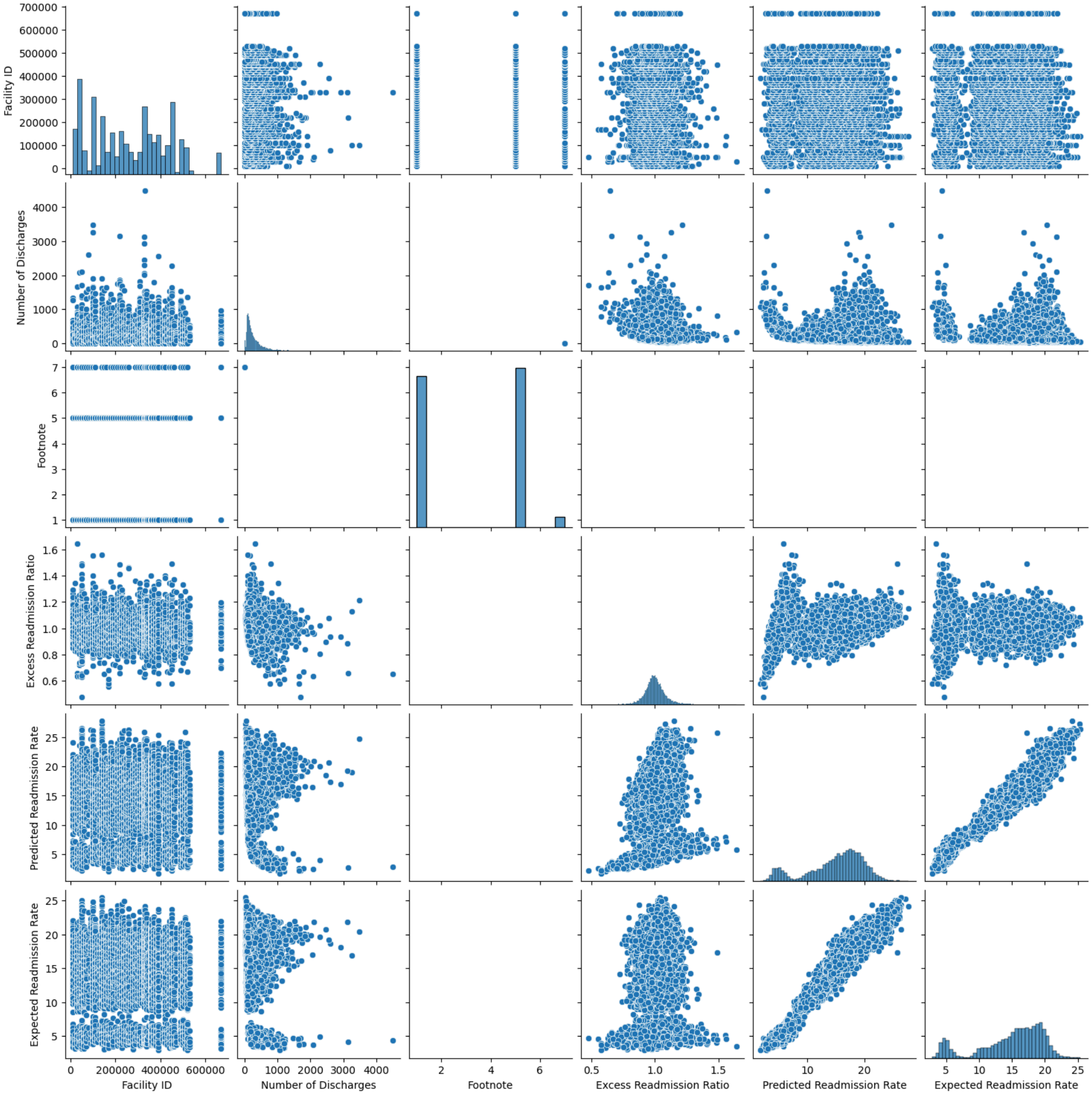
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

hospital = pd.read_csv("FY_2025_Hospital_Readmissions_Reduction_Program_Hospital.cs

hospital.describe()
#hospital.columns
```

Out[1]:

	Facility ID	Number of Discharges	Footnote	Excess Readmission Ratio	Predicted Readmission Rate	Expected Readmission Rate
count	18510.000000	8340.000000	6583.000000	11927.000000	11927.000000	11927.000000
mean	261770.055105	279.269904	3.187756	1.001719	14.995386	14.961234
std	164647.739172	266.018069	2.089167	0.080547	5.017854	4.871997
min	10001.000000	0.000000	1.000000	0.477900	1.674200	2.892100
25%	110073.000000	115.000000	1.000000	0.956550	12.533000	12.612800
50%	250048.000000	197.000000	5.000000	0.998200	16.060200	16.146000
75%	390133.000000	354.000000	5.000000	1.043000	18.609000	18.667350
max	670327.000000	4501.000000	7.000000	1.643000	27.809500	25.394200

In [2]:

```
# Stole this from my HW 3
sns.pairplot(hospital)
plt.show()
```



In [3]:

```
# Noticed footnote so I wanna check for nulls (stole from HW 3 as well)
print(hospital.isnull().sum())
```

```
Facility Name                     0
Facility ID                       0
State                             0
Measure Name                      0
Number of Discharges          10170
Footnote                      11927
Excess Readmission Ratio       6583
Predicted Readmission Rate     6583
Expected Readmission Rate      6583
Number of Readmissions         6583
Start Date                        0
End Date                          0
dtype: int64
```

In [4]:

```
print(f'Total Columns: {len(hospital)}')
print (hospital["Footnote"].isnull().sum() + hospital["Excess Readmission Ratio"].i
# So Footnote and excess Readmission Ratio, Predicted Readmission Rate, Expected Re
```

```
Total Columns: 18510
18510
```

(5 points): Describe each of the 12 variables in your own words, then mention the datatype of each.

In [5]:

```
hospital.dtypes
```

Out[5]:

```
Facility Name                  object
Facility ID                     int64
State                          object
Measure Name                   object
Number of Discharges          float64
Footnote                      float64
Excess Readmission Ratio      float64
Predicted Readmission Rate    float64
Expected Readmission Rate     float64
Number of Readmissions         object
Start Date                     object
End Date                       object
dtype: object
```

The 12 columns, what they are, and their data types are as follow:

1. Facility Name, the name of the hospital this patient was under study for, this is an object type (category since its just a string).
2. Facility ID is the ID of the hospital encoded to its name, the dtype is int64 but it should probably be treated as a categorical variable as well.
3. State; the state the hospital is in; object since its 2 letters.
4. Measure Name; I think this is the condition the data pertains to?; object since its strings.
5. Number of Discharges; # of discharges for the measure/condition; float64 since its a number.
6. Footnote; I think this is a reason for not including any of the following 4 columns; float64 since its a number.
7. Excess Readmission Ratio; a ratio of expected readmissions > 1 is more than expected and vice versa for the condition, float64 since its a number.
8. Predicted Readmission Rate; predicted readmissions for the condition * the readmission rate for patient/condition. float64 since num.
9. Expected Readmission Rate; this is like the national avg for the condition or patient or smth like that. float64 since num.
10. Number of Readmissions; the num of readmissions for that condition, object since a lot of data points are "Too Few to Report"
11. Start Date; beginning of data collection, object since its structured date format.
12. End Date; end of data collection, object since its structured date format.

(5 points): Form your research question that can be answered by this dataset.

What factors impact the number of readmissions the most? By extension: Which and how can these factors be manipulated to reduce the number of reradmissions.

(5 points): Explain why your research question would be interesting to the board-- do not tailor your research question to me just because I'm your machine learning instructor. I'm interested in your model, but the board cares about money/patients

The board most likely cares about saving money more than anything else, so if I taylor my question to help answer how to reduce readmissions this would be enticing for them.

(40 points): Choose any algorithm from chapter 5, 6, 7, or 8 to answer your research question. Explain your choice.

```
Write your algorithm from scratch.
Include resources used for writing your algorithm.
If you choose to use generative AI-- and the gen AI model gets
something wrong -- you will be docked for its mistakes. A mistake
can include, but is not limited to: code mistakes, getting the
right answer for the wrong reason, using a model for the wrong
datatypes, ethical violations and assumptions made by the model,
etc...
```

A: I am going to be doing a stepwise forward selection to predict the number of readmissions for my model. This will kind of know out the bad features and also give coefficients for the more important features.

Here are my sources:

https://automaticaddison.com/stepwise-forward-selection-algorithm-from-scratch/

https://fakhredin.medium.com/forward-selection-to-find-predictive-variables-with-python-code-3c33f0db2393

https://en.wikipedia.org/wiki/Coefficient_of_determination

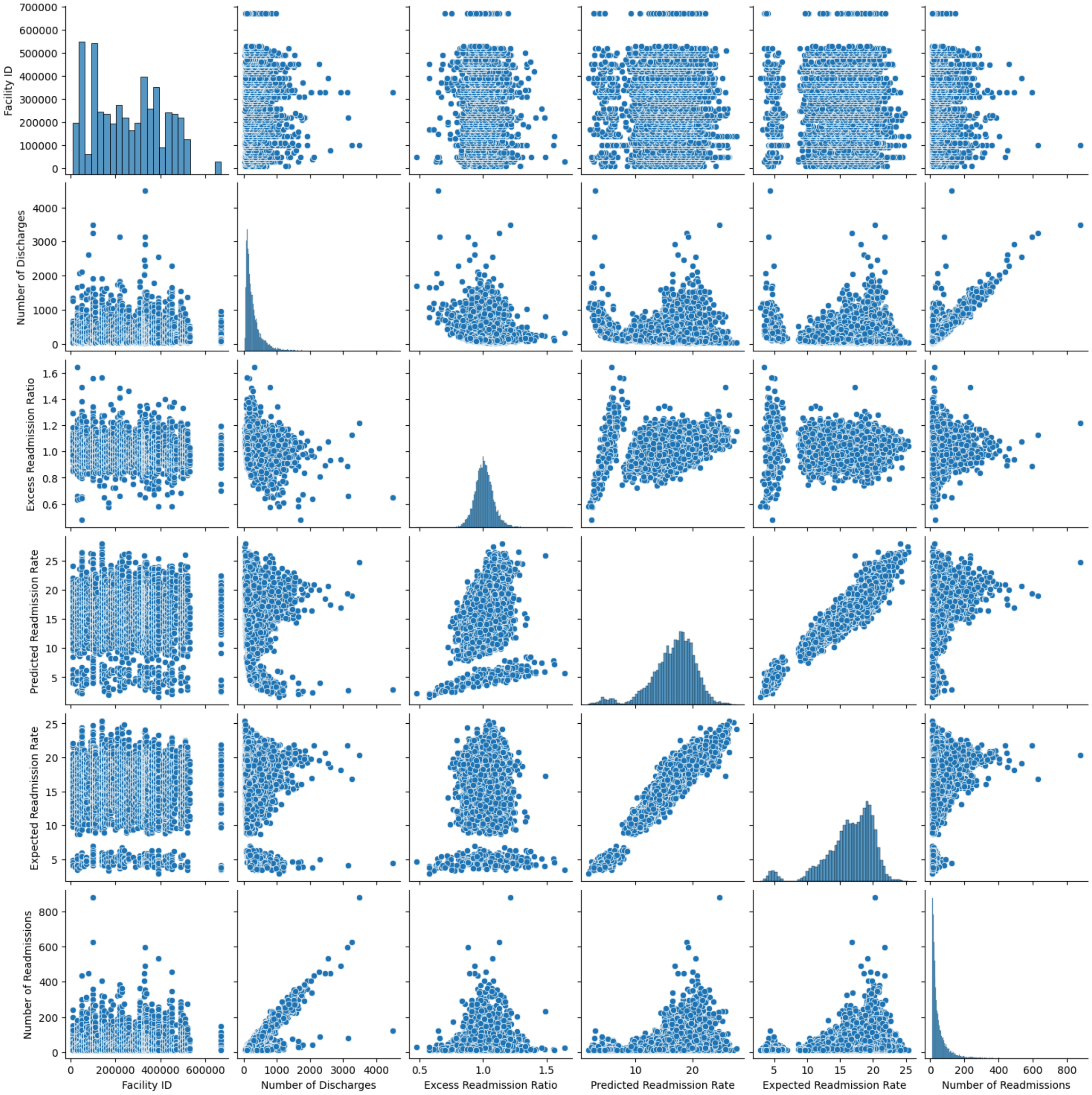
In [6]:

```
# First I wanna clean up my dataset (drop footnote first)
hospital.drop(['Footnote'], axis=1, inplace=True)  # https://pandas.pydata.org/pand
hospital.dropna(subset = ['Number of Readmissions'], inplace=True) # https://www.st
# I think I have enough data to drop na num of discharges:
hospital.dropna(subset = ['Number of Discharges'], inplace=True) # https://www.stat
print(hospital.isnull().sum())
```

```
Facility Name                 0
Facility ID                   0
State                         0
Measure Name                  0
Number of Discharges          0
Excess Readmission Ratio      0
Predicted Readmission Rate    0
Expected Readmission Rate     0
Number of Readmissions        0
Start Date                    0
End Date                      0
dtype: int64
```

In [7]:

```
hospital.dtypes
hospital["Number of Readmissions"] = hospital["Number of Readmissions"].replace("To
hospital["Number of Readmissions"] = pd.to_numeric(hospital["Number of Readmissions

# I basically turned the readmissions that are too few into 0 so I can include them
sns.pairplot(hospital)
plt.show()
```



In [8]:

```
# My code for linear reg from hw 3: (I had to add the r_squared feature to work wit
class MyLinearReg():
    # Here I am creating a constructor for my class
    def __init__(self, learning_rate, it):
        self.learning_rate = learning_rate # This if for gradient descent
        self.it = it # This is for gradient decent

    # This trains the model
    def fit(self, X, Y):
        # Initializing X and Y
        X = np.array(X)
        Y = np.array(Y)

        # n is the number of features (2)
        self.m, self.n = X.shape

        # Initializing my weight
        self.B = np.zeros(self.n) # Modified this to allow for more B
        self.b_0 = 0
        self.X = X
        self.Y = Y

        # This is the gradient decent
        for i in range(self.it):
            self.update_weights()
        return self

    # This is the implementation of gradient decent formula
    def update_weights(self):
        Y_pred = self.predict(self.X)
        dB = -(2 * (self.X.T).dot(self.Y - Y_pred)) / self.m
        db_0 = -2 * np.sum(self.Y - Y_pred) / self.m

        # Here the weight is adjusted according to the learning rate.
        self.B = self.B - self.learning_rate * dB
        self.b_0 = self.b_0 - self.learning_rate * db_0

        return self

    def predict(self, X):
        return X.dot(self.B) + self.b_0

    def describe(self):
        print("The values of B is:", self.B)
        print("The value of B_0 is: ", self.b_0)

    # I had to add this for the forward stepwise, i just used the r^2 formula we ha
    # also used this for formula: https://en.wikipedia.org/wiki/Coefficient_of_dete
    def r_squared(self):
        y_pred = self.predict(self.X)
        ss_res = np.sum((self.Y - y_pred) ** 2)
        ss_tot = np.sum((self.Y - np.mean(self.Y)) ** 2)
        r2 = 1 - (ss_res / ss_tot)
        return r2
```

In [9]:

```
# need to drop all non-numeric
print(hospital.dtypes)
hospital.drop(['Facility Name', 'Facility ID', 'Facility Name', 'State', 'Start Dat
```

```
Facility Name                  object
Facility ID                     int64
State                          object
Measure Name                   object
Number of Discharges          float64
Excess Readmission Ratio      float64
Predicted Readmission Rate    float64
Expected Readmission Rate     float64
Number of Readmissions          int64
Start Date                     object
End Date                       object
dtype: object
```

In [10]:

```
# I want to use measure name still so Ill encode it
# SRC: https://stackoverflow.com/questions/37292872/how-can-i-one-hot-encode-in-pyt
# encoded = pd.get_dummies(hospital['Measure Name'], prefix='Measure').astype(int)
# hospital = pd.concat([hospital.drop('Measure Name', axis=1), encoded], axis=1)
# print(hospital.dtypes)

### NOTE: nevermind, it broke my code
```

In [11]:

```
# Actual algorithm start: Everything is breaking for some reason :(
ar2 = dict()
candidates = []
last_max = -1

y = 'Number of Readmissions'

while(True):
    for x in hospital.drop([y] + candidates, axis=1).columns:
        if len(candidates) == 0:
            features = [x]
        else:
            features = [x] + candidates

        model = MyLinearReg(it=10000, learning_rate=0.000001)
        model.fit(hospital[features], hospital[y])
        ar2[x] = model.r_squared()

    max_ar2 =  max(ar2.values())
    max_ar2_key = max(ar2, key=ar2.get)

    if max_ar2 > last_max:
        candidates.append(max_ar2_key)
        last_max = max_ar2

        print('step: ' + str(len(candidates)))
        print(candidates)
        print('Adjusted R2: ' + str(max_ar2))
        print('===============')
    else:
        print(model.describe())
        break

print('\n\n')
print('eliminated variables: ')
print(set(hospital.drop(y, axis=1).columns).difference(candidates))
```

```
step: 1
['Number of Discharges']
Adjusted R2: 0.7992126273489623
===============
step: 2
['Number of Discharges', 'Predicted Readmission Rate']
Adjusted R2: 0.8134755171678694
===============
step: 3
['Number of Discharges', 'Predicted Readmission Rate', 'Excess Readmission Ratio']
Adjusted R2: 0.8135228933312568
===============
The values of B is: [ 0.12791634  0.15164401  0.32456306 -0.03103499]
The value of B_0 is:  -0.04562451538144902
None



eliminated variables:
{'Expected Readmission Rate'}
```

In [12]:

```python
# Fitting model with everything but Expected Readmission Rate:
X = hospital[['Number of Discharges', 'Predicted Readmission Rate', 'Excess Readmis
Y = hospital['Number of Readmissions']

model = MyLinearReg(learning_rate=0.000001, it=10000)

model.fit(X, Y)

model.describe()
model.r_squared()
```

```
The values of B is: [ 0.15149975  0.45589187 -0.02384048]
The value of B_0 is:  -0.03843306681496902
```

Out[12]:

```
np.float64(0.8135228933312568)
```

(15 points): You need to validate your algorithm! If you want to use a package for this, that is okay.

In [13]:

```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

model_sklearn = LinearRegression()
model_sklearn.fit(X, Y)

# Calculate coefficients and r squared
print("sklearn coefficient (B):", model_sklearn.coef_)
print("sklearn intercept (B_0):", model_sklearn.intercept_)

Y_pred_sklearn = model_sklearn.predict(X)

r2_sklearn = r2_score(Y, Y_pred_sklearn)

print(f"sklearn R-squared: {r2_sklearn}")
```

```
sklearn coefficient (B): [ 0.16794386  3.56523232 86.95667148]
sklearn intercept (B_0): -147.3791049396031
sklearn R-squared: 0.9197903563760922
```

As shown, the coefficient of my models are pretty similar:

My model: The values of B is: [0.15149975 0.45589187 -0.02384048] The value of B_0 is: -0.03843306681496902 R Squared is: 0.8135228933312568

SKLearn: sklearn coefficient (B): [0.16794386 3.56523232 86.95667148] sklearn intercept (B_0): -147.3791049396031 sklearn R-squared: 0.9197903563760922

The R squared values are very similar but the coefficients are quite different (especially the last). I think I had to scale but everytime I tried everything broe so I ended up giving up on that.

(5 points): In a few sentences, tell the board your conclusions, predictions, and recommendations.

In conclusion, the most important factors are Number of Discharges, Predicted Readmission Rate, and Excess Readmission Ratio, all of which positively affect the amount of readmissions. Surprisingly, Expected Readmission Rate is not as impactful in the model. As such, I would recommend trying to reduce these ratios?