Estimating Treatment Effect Using Machine Learning

Welcome to the first assignment of **AI for Medical Treatment**!

You will be using different methods to evaluate the results of a randomized control trial (RCT).

You will learn:

- How to analyze data from a randomized control trial using both:
  - traditional statistical methods
  - and the more recent machine learning techniques
- Interpreting Multivariate Models
  - Quantifying treatment effect
  - Calculating baseline risk
  - Calculating predicted risk reduction
- Evaluating Treatment Effect Models
  - Comparing predicted and empirical risk reductions
  - Computing C-statistic-for-benefit
- Interpreting ML models for Treatment Effect Estimation
  - Implement T-learner

This assignment covers the folowing topics:

Packages

We'll first import all the packages that we need for this assignment.

- `pandas` is what we'll use to manipulate our data
- `numpy` is a library for mathematical and scientific operations
- `matplotlib` is a plotting library
- `sklearn` contains a lot of efficient tools for machine learning and statistical modeling
- `random` allows us to generate random numbers in python
- `lifelines` is an open-source library that implements c-statistic
- `itertools` will help us with hyperparameters searching

Import Packages

Run the next cell to import all the necessary packages, dependencies and custom util functions.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sklearn
import random
import lifelines
import itertools

plt.rcParams['figure.figsize'] = [10, 7]
```

1 Dataset

1.1 Why RCT?

In this assignment, we'll be examining data from an RCT, measuring the effect of a particular drug combination on colon cancer. Specifically, we'll be looking the effect of Levamisole and Fluorouracil on patients who have had surgery to remove their colon cancer. After surgery, the curability of the patient depends on the remaining residual cancer. In this study, it was found that this particular drug combination had a clear beneficial effect, when compared with Chemotherapy.

1.2 Data Processing

In this first section, we will load in the dataset and calculate basic statistics. Run the next cell to load the dataset. We also do some preprocessing to convert categorical features to one-hot representations.

In [2]:
```
data = pd.read_csv("levamisole_data.csv", index_col=0)
```

Let's look at our data to familiarize ourselves with the various fields.

In [3]:
```
print(f"Data Dimensions: {data.shape}")
data.head()
```

```
Data Dimensions: (607, 14)
```

Out[3]:

	sex	age	obstruct	perfor	adhere	nodes	node4	outcome	TRTMT	differ_2.0	differ_3.0	extent_2	extent_3	extent_4
1	1	43	0	0	0	5.0	1	1	True	1	0	0	1	0
2	1	63	0	0	0	1.0	0	0	True	1	0	0	1	0
3	0	71	0	0	1	7.0	1	1	False	1	0	1	0	0
4	0	66	1	0	0	6.0	1	1	True	1	0	0	1	0
5	1	69	0	0	0	22.0	1	1	False	1	0	0	1	0

Below is a description of all the fields (one-hot means a different field for each level):

- `sex (binary): 1 if Male, 0 otherwise`
- `age (int): age of patient at start of the study`
- `obstruct (binary): obstruction of colon by tumor`
- `perfor (binary): perforation of colon`
- `adhere (binary): adherence to nearby organs`
- `nodes (int): number of lymphnodes with detectable cancer`
- `node4 (binary): more than 4 positive lymph nodes`
- `outcome (binary): 1 if died within 5 years`
- `TRTMT (binary): treated with levamisole + fluorouci`l
- `differ (one-hot): differentiation of tumor`
- `extent (one-hot): extent of local spread`

In particular pay attention to the `TRTMT` and `outcome` columns. Our primary endpoint for our analysis will be the 5-year survival rate, which is captured in the `outcome` variable.

Exercise 01

Since this is an RCT, the treatment column is randomized. Let's warm up by finding what the treatment probability is.

$$p_{treatment} = \frac{n_{treatment}}{n}$$

- $n_{treatment}$ is the number of patients where `TRTMT = True`
- n is the total number of patients.

In [4]:
```
# UNQ_C1 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def proportion_treated(df):
    """
    Compute proportion of trial participants who have been treated

    Args:
        df (dataframe): dataframe containing trial results. Column
                        'TRTMT' is 1 if patient was treated, 0 otherwise.

    Returns:
        result (float): proportion of patients who were treated
    """

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    proportion = np.sum(df.TRTMT)/len(df.TRTMT)

    ### END CODE HERE ###

    return proportion
```

Test Case

In [5]:
```
print("dataframe:\n")
example_df = pd.DataFrame(data =[[0, 0],
                                 [1, 1],
                                 [1, 1],
                                 [1, 1]], columns = ['outcome', 'TRTMT'])
print(example_df)
print("\n")
treated_proportion = proportion_treated(example_df)
print(f"Proportion of patient treated: computed {treated_proportion}, expected: 0.75")
```

```
dataframe:

   outcome  TRTMT
0        0      0
1        1      1
2        1      1
3        1      1


Proportion of patient treated: computed 0.75, expected: 0.75
```

Next let's run it on our trial data.

In [6]:
```
p = proportion_treated(data)
print(f"Proportion Treated: {p} ~ {int(p*100)}%")
```

```
Proportion Treated: 0.49093904448105435 ~ 49%
```

Exercise 02

Next, we can get a preliminary sense of the results by computing the empirical 5-year death probability for the treated arm versus the control arm.

The probability of dying for patients who received the treatment is:

$$p_{\text{treatment, death}} = \frac{n_{\text{treatment,death}}}{n_{\text{treatment}}}$$

- $n_{\text{treatment,death}}$ is the number of patients who received the treatment and died.
- $n_{\text{treatment}}$ is the number of patients who received treatment.

The probability of dying for patients in the control group (who did not received treatment) is:

$$p_{\text{control, death}} = \frac{n_{\text{control,death}}}{n_{\text{control}}}$$

- $n_{\text{control,death}}$ is the number of patients in the control group (did not receive the treatment) who died.
- n_{control} is the number of patients in the control group (did not receive treatment).

In [7]:
```
# UNQ_C2 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def event_rate(df):
    '''
    Compute empirical rate of death within 5 years
    for treated and untreated groups.

    Args:
        df (dataframe): dataframe containing trial results.
                        'TRTMT' column is 1 if patient was treated, 0 otherwise.
                         'outcome' column is 1 if patient died within 5 years, 0 otherwise.

    Returns:
        treated_prob (float): empirical probability of death given treatment
        untreated_prob (float): empirical probability of death given control
    '''

    treated_prob = 0.0
    control_prob = 0.0

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    treated_prob = np.sum(df.outcome & df.TRTMT)/np.sum(df.TRTMT)
    control_prob = np.sum(df.outcome & (1-df.TRTMT))/np.sum(1-df.TRTMT)

    ### END CODE HERE ###

    return treated_prob, control_prob
```

Test Case

In [8]:
```
print("TEST CASE\ndataframe:\n")
example_df = pd.DataFrame(data =[[0, 1],
                                 [1, 1],
                                 [1, 1],
                                 [0, 1],
                                 [1, 0],
                                 [1, 0],
                                 [1, 0],
                                 [0, 0]], columns = ['outcome', 'TRTMT'])
#print("dataframe:\n")
print(example_df)
print("\n")
treated_prob, control_prob = event_rate(example_df)
print(f"Treated 5-year death rate, expected: 0.5, got: {treated_prob:.4f}")
print(f"Control 5-year death rate, expected: 0.75, got: {control_prob:.4f}")
```

```
TEST CASE
dataframe:

   outcome  TRTMT
0        0      1
1        1      1
2        1      1
3        0      1
4        1      0
5        1      0
6        1      0
7        0      0


Treated 5-year death rate, expected: 0.5, got: 0.5000
Control 5-year death rate, expected: 0.75, got: 0.7500
```

Now let's try the function on the real data.

In [9]:
```
treated_prob, control_prob = event_rate(data)

print(f"Death rate for treated patients: {treated_prob:.4f} ~ {int(treated_prob*100)}%")
print(f"Death rate for untreated patients: {control_prob:.4f} ~ {int(control_prob*100)}%")
```

```
Death rate for treated patients: 0.3725 ~ 37%
Death rate for untreated patients: 0.4822 ~ 48%
```

On average, it seemed like treatment had a positive effect.

Sanity checks

It's important to compute these basic summary statistics as a sanity check for more complex models later on. If they strongly disagree with these robust summaries and there isn't a good reason, then there might be a bug.

Train test split

We'll now try to quantify the impact more precisely using statistical models. Before we get started fitting models to analyze the data, let's split it using the `train_test_split` function from `sklearn`. While a hold-out test set isn't required for logistic regression, it will be useful for comparing its performance to the ML models later on.

In [10]:
```
# As usual, split into dev and test set
from sklearn.model_selection import train_test_split
np.random.seed(18)
random.seed(1)

data = data.dropna(axis=0)
y = data.outcome
# notice we are dropping a column here. Now our total columns will be 1 less than before
X = data.drop('outcome', axis=1)
X_dev, X_test, y_dev, y_test = train_test_split(X, y, test_size = 0.25, random_state=0)
```

In [11]:
```
print(f"dev set shape: {X_dev.shape}")
print(f"test set shape: {X_test.shape}")
```

```
dev set shape: (455, 13)
test set shape: (152, 13)
```

2 Modeling Treatment Effect

2.1 Constant Treatment Effect

First, we will model the treatment effect using a standard logistic regression. If $x^{(i)}$ is the input vector, then this models the probability of death within 5 years as

$$\sigma(\theta^T x^{(i)}) = \frac{1}{1 + exp(-\theta^T x^{(i)})},$$

where $\theta^T x^{(i)} = \sum_j \theta_j x_j^{(i)}$ is an inner product.

For example, if we have three features, $TRTMT$, AGE, and SEX, then our probability of death would be written as:

$$\sigma(\theta^T x^{(i)}) = \frac{1}{1 + exp(-\theta_{TRTMT} x_{TRTMT}^{(i)} - \theta_{AGE} x_{AGE}^{(i)} - \theta_{SEX} x_{SEX}^{(i)})}.$$

Another way to look at logistic regresion is as a linear model for the "logit" function, or "log odds":

$$logit(p) = \log\left(\frac{p}{1-p}\right) = \theta^T x^{(i)}$$

- "Odds" is defined as the probability of an event divided by the probability of not having the event: $\frac{p}{1-p}$.
- "Log odds", or "logit" function, is the natural log of the odds: $log\left(\frac{p}{1-p}\right)$

In this example, $x_{TRTMT}^{(i)}$ is the treatment variable. Therefore, θ_{TRTMT} tells you what the effect of treatment is. If θ_{TRTMT} is negative, then having treatment reduces the log-odds of death, which means death is less likely than if you did not have treatment.

Note that this assumes a constant relative treatment effect, since the impact of treatment does not depend on any other covariates.

Typically, a randomized control trial (RCT) will seek to establish a negative θ_{TRTMT} (because the treatment is intended to reduce risk of death), which corresponds to an odds ratio of less than 1.

An odds ratio of less than one implies the probability of death is less than the probability of surviving.

$$\frac{p}{1-p} < 1 \rightarrow p < 1 - p$$

Run the next cell to fit your logistic regression model.

You can use the entire dev set (and do not need to reserve a separate validation set) because there is no need for hyperparameter tuning using a validation set.

In [12]:
```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(penalty='l2',solver='lbfgs', max_iter=10000).fit(X_dev, y_dev)
```

Calculating the Odds ratio

You are interested in finding the odds for treatment relative to the odds for the baseline.

$$OddsRatio = \frac{Odds_{treatment}}{Odds_{baseline}}$$

where

$$Odds_{treatment} = \frac{p_{treatment}}{1 - p_{treatment}}$$

and

$$Odds_{baseline} = \frac{p_{baseline}}{1 - p_{baseline}}$$

If you look at the expression

$$\log\left(\frac{p}{1-p}\right) = \theta^T x^{(i)} = \theta_{treatment} \times x_{treatment}^{(i)} + \theta_{age} \times x_{age}^{(i)} + \cdots$$

Let's just let "$\theta \times x_{age}^{(i)} + \cdots$" stand for all the other thetas and feature variables except for the treatment $\theta_{treatment}^{(i)}$, and $x_{treatment}^{(i)}$.

Treatment

To denote that the patient received treatment, we set $x_{treatment}^{(i)} = 1$. Which means the log odds for a treated patient are:

$$log(Odds_{treatment}) = \log\left(\frac{p_{treatment}}{1 - p_{treatment}}\right) = \theta_{treatment} \times 1 + \theta_{age} \times x_{age}^{(i)} + \cdots$$

To get odds from log odds, use exponentiation (raise to the power of e) to take the inverse of the natural log.

$$Odds_{treatment} = e^{log(Odds_{treatment})} = \left(\frac{p_{treatment}}{1 - p_{treatment}}\right) = e^{\theta_{treatment} \times 1 + \theta_{age} \times x_{age}^{(i)} + \cdots}$$

Control (baseline)

Similarly, when the patient has no treatment, this is denoted by $x_{treatment}^{(i)} = 0$. So the log odds for the untreated patient is:

$$log(Odds_{baseline}) = \log\left(\frac{p_{baseline}}{1 - p_{baseline}}\right) = \theta_{treatment} \times 0 + \theta_{age} \times x_{age}^{(i)} + \cdots$$
$$= 0 + \theta_{age} \times x_{age}^{(i)} + \cdots$$

To get odds from log odds, use exponentiation (raise to the power of e) to take the inverse of the natural log.

$$Odds_{baseline} = e^{log(Odds_{baseline})} = \left(\frac{p_{baseline}}{1 - p_{baseline}}\right) = e^{0 + \theta_{age} \times x_{age}^{(i)} + \cdots}$$

Odds Ratio

The Odds ratio is:

$$OddsRatio = \frac{Odds_{treatment}}{Odds_{baseline}}$$

Doing some substitution:

$$OddsRatio = \frac{e^{\theta_{treatment} \times 1 + \theta_{age} \times x_{age}^{(i)} + \cdots}}{e^{0 + \theta_{age} \times x_{age}^{(i)} + \cdots}}$$

Notice that $e^{\theta_{age} \times x_{age}^{(i)} + \cdots}$ cancels on top and bottom, so that:

$$OddsRatio = \frac{e^{\theta_{treatment} \times 1}}{e^0}$$

Since $e^0 = 1$, This simplifies to:

$$OddsRatio = e^{\theta_{treatment}}$$

Exercise 03: Extract the treatment effect

Complete the `extract_treatment_effect` function to extract $\theta_{treatment}$ and then calculate the odds ratio of treatment from the logistic regression model.

In [13]:
```
# UNQ_C3 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def extract_treatment_effect(lr, data):
    theta_TRTMT = 0.0
    TRTMT_OR = 0.0
    coeffs = {data.columns[i]:lr.coef_[0][i] for i in range(len(data.columns))}

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    # get the treatment coefficient
    theta_TRTMT = coeffs["TRTMT"]

    # calculate the Odds ratio for treatment
    TRTMT_OR = np.exp(theta_TRTMT)

    ### END CODE HERE ###
    return theta_TRTMT, TRTMT_OR
```

Test

In [14]:
```
# Test extract_treatment_effect function
theta_TRTMT, trtmt_OR = extract_treatment_effect(lr, X_dev)
print(f"Theta_TRTMT: {theta_TRTMT:.4f}")
print(f"Treatment Odds Ratio: {trtmt_OR:.4f}")
```

```
Theta_TRTMT: -0.2885
Treatment Odds Ratio: 0.7494
```

Expected Output

```
Theta_TRTMT: -0.2885
Treatment Odds Ratio: 0.7494
```

Based on this model, it seems that the treatment has a beneficial effect.

- The $\theta_{treatment} = -0.29$ is a negative value, meaning that it has the effect of reducing risk of death.
- In the code above, the $OddsRatio$ is stored in the variable `TRTMT_OR`.
- The $OddsRatio = 0.75$, which is less than 1.

You can think of the $OddsRatio$ as a factor that is multiplied to the baseline odds $Odds_{baseline}$ in order to estimate the $Odds_{treatment}$. You can think about the Odds Ratio as a rate, converting between baseline odds and treatment odds.

$$Odds_{treatment} = OddsRatio \times Odds_{baseline}$$

In this case:

$$Odds_{treatment} = 0.75 \times Odds_{baseline}$$

So you can interpret this to mean that the treatment reduces the odds of death by $(1 - OddsRatio) = 1 - 0.75 = 0.25$, or about 25%.

You will see how well this model fits the data in the next few sections.

2.2 Absolute Risk Reduction

Exercise 4: Calculate ARR

A valuable quantity is the absolute risk reduction (ARR) of a treatment. If p is the baseline probability of death, and $p_{treatment}$ is the probability of death if treated, then

$$ARR = p_{baseline} - p_{treatment}$$

In the case of logistic regression, here is how ARR can be computed:
Recall that the Odds Ratio is defined as:

$$OR = Odds_{treatment}/Odds_{baseline}$$

where the "odds" is the probability of the event over the probability of not having the event, or $p/(1-p)$.

$$Odds_{trtmt} = \frac{p_{treatment}}{1 - p_{treatment}}$$

and

$$Odds_{baseline} = \frac{p_{baseline}}{1 - p_{baseline}}$$

In the function below, compute the predicted absolute risk reduction (ARR) given

- the odds ratio for treatment "OR", and
- the baseline risk of an individual $p_{baseline}$

If you get stuck, try reviewing the level 1 hints by clicking on the cell "Hints Level 1". If you would like more help, please try viewing "Hints Level 2".

Hints Level 1

- Using the given p, compute the baseline odds of death.
- Then, use the Odds Ratio to convert that to odds of death given treatment.
- Finally, convert those odds back into a probability

Hints Level 2

- Solve for p_treatment starting with this expression: Odds_treatment = p_treatment / (1 - p_treatment). You may want to do this on a piece of paper.

In [15]:
```
# UNQ_C4 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def OR_to_ARR(p, OR):
    """
    Compute ARR for treatment for individuals given
    baseline risk and odds ratio of treatment.
```

```
    Args:
        p (float): baseline probability of risk (without treatment)
        OR (float): odds ratio of treatment versus baseline

    Returns:
        ARR (float): absolute risk reduction for treatment
      """

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    # compute baseline odds from p
    odds_baseline = p/(1-p)

    # compute odds of treatment using odds ratio
    odds_trtmt = OR*odds_baseline

    # compute new probability of death from treatment odds
    p_trtmt = odds_trtmt/(odds_trtmt+1)

    # compute ARR using treated probability and baseline probability
    ARR = p-p_trtmt

    ### END CODE HERE ###

    return ARR
```

Test Case

In [16]:
```
print("TEST CASES")
test_p, test_OR = (0.75, 0.5)
print(f"baseline p: {test_p}, OR: {test_OR}")
print(f"Output: {OR_to_ARR(test_p, test_OR):.4f}, Expected: {0.15}\n")

test_p, test_OR = (0.04, 1.2)
print(f"baseline p: {test_p}, OR: {test_OR}")
print(f"Output: {OR_to_ARR(test_p, test_OR):.4f}, Expected: {-0.0076}")
```

```
TEST CASES
baseline p: 0.75, OR: 0.5
Output: 0.1500, Expected: 0.15

baseline p: 0.04, OR: 1.2
Output: -0.0076, Expected: -0.0076
```

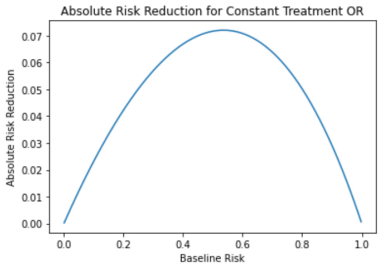
Visualize the treatment effect as baseline risk varies

The logistic regression model assumes that treatment has a constant effect in terms of odds ratio and is independent of other covariates.

However, this does not mean that absolute risk reduction is necessarily constant for any baseline risk $\hat{p}$. To illustrate this, we can plot absolute risk reduction as a function of baseline predicted risk $\hat{p}$.

Run the next cell to see the relationship between ARR and baseline risk for the logistic regression model.

In [17]:
```
ps = np.arange(0.001, 0.999, 0.001)
diffs = [OR_to_ARR(p, trtmt_OR) for p in ps]
plt.plot(ps, diffs)
plt.title("Absolute Risk Reduction for Constant Treatment OR")
plt.xlabel('Baseline Risk')
plt.ylabel('Absolute Risk Reduction')
plt.show()
```



Note that when viewed on an absolute scale, the treatment effect is not constant, despite the fact that you used a model with no interactions between the features (we didn't multiply two features together).

As shown in the plot, when the baseline risk is either very low (close to zero) or very high (close to one), the Absolute Risk Reduction from treatment is fairly low. When the baseline risk is closer to 0.5 the ARR of treatment is higher (closer to 0.10).

It is always important to remember that baseline risk has a natural effect on absolute risk reduction.

2.3 Model Limitations

We can now plot how closely the empirical (actual) risk reduction matches the risk reduction that is predicted by the logistic regression model.

This is complicated by the fact that for each patient, we only observe one outcome (treatment or no treatment).

- We can't give a patient treatment, then go back in time and measure an alternative scenario where the same patient did not receive the treatment.
- Therefore, we will group patients into groups based on their baseline risk as predicted by the model, and then plot their empirical ARR within groups that have similar baseline risks.
- The empirical ARR is the death rate of the untreated patients in that group minus the death rate of the treated patients in that group.

$$ARR_{empirical} = p_{baseline} - p_{treatment}$$

Exercise 5: Baseline Risk

In the next cell, write a function to compute the baseline risk of each patient using the logistic regression model.

The baseline risk is the model's predicted probability that the patient is predicted to die if they do not receive treatment.

You will later use the baseline risk of each patient to organize patients into risk groups (that have similar baseline risks). This will allow you to calculate the ARR within each risk group.

$$p_{baseline} = logisticRegression(Treatment = False, Age = age_i, Obstruct = obstruct_i, \cdots)$$

Hints

- A patient receives treatment if their feature x_treatment is True, and does not receive treatment when their x_treatment is False.
- For a patient who actually did receive treatment, you can ask the model to predict their risk without receiving treatment by setting the patient's x_treatment to False.
- The logistic regression predict_proba() function returns a 2D array, one row for each patient, and one column for each possible outcome (each class). In this case, the two outcomes are either no death (0), or death (1). To find out which column contains the probability for death, check the order of the classes by using lr.classes_

In [18]:
```
# UNQ_C5 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def base_risks(X, lr_model):
    """
    Compute baseline risks for each individual in X.

    Args:
        X (dataframe): data from trial. 'TRTMT' column
                       is 1 if subject retrieved treatment, 0 otherwise
        lr_model (model): logistic regression model

    Returns:
        risks (np.array): array of predicted baseline risk
                          for each subject in X
    """

    # first make a copy of the dataframe so as not to overwrite the original
    X = X.copy(deep=True)

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###

    # Set the treatment variable to assume that the patient did not receive treatment
    for i in range(len(X.TRTMT)):
        X.TRTMT[i]=0

    # Input the features into the model, and predict the probability of death.
    risks = lr_model.predict_proba(X)[:,1]

    # END CODE HERE

    return risks
```

Test Case

In [19]:
```
example_df = pd.DataFrame(columns = X_dev.columns)
example_df.loc[0, :] = X_dev.loc[X_dev.TRTMT == 1, :].iloc[0, :]
example_df.loc[1, :] = example_df.iloc[0, :]
example_df.loc[1, 'TRTMT'] = 0

print("TEST CASE")
print(example_df)
print(example_df.loc[:, ['TRTMT']])
print('\n')

print("Base risks for both rows should be the same")
print(f"Baseline Risks: {base_risks(example_df.copy(deep=True), lr)}")
```

```
TEST CASE
  sex age obstruct perfor adhere nodes node4 TRTMT differ_2.0 differ_3.0  \
0   1  60        0      0      0     3     0  True          1          0
1   1  60        0      0      0     3     0     0          1          0

  extent_2 extent_3 extent_4
0        0        1        0
1        0        1        0
  TRTMT
0  True
1     0


Base risks for both rows should be the same
Baseline Risks: [0.43115868 0.43115868]
```

Expected output

```
Base risks for both rows should be the same
Baseline Risks: [0.43115868 0.43115868]
```

Exercise 6: ARR by quantile

Since the effect of treatment varies depending on the baseline risk, it makes more sense to group patients who have similar baseline risks, and then look at the outcomes of those who receive treatment versus those who do not, to estimate the absolute risk reduction (ARR).

You'll now implement the `lr_ARR_quantile` function to plot empirical average ARR for each quantile of base risk.

Hints

- Use pandas.cut to define intervals of bins of equal size. For example, pd.cut(arr,5) uses the values in the list or array 'arr' and returns the intervals of 5 bins.
- Use pandas.DataFrame.groupby to group by a selected column of the dataframe. Then select the desired variable and apply an aggregator function. For example, df.groupby('col1')['col2'].sum() groups by column 1, and then calculates the sum of column 2 for each group.

In [20]:
```
# UNQ_C6 (UNIQUE CELL IDENTIFIER, DO NOT EDIT)
def lr_ARR_quantile(X, y, lr):

    # first make a deep copy of the features dataframe to calculate the base risks
    X = X.copy(deep=True)

    # Make another deep copy of the features dataframe to store baseline risk, risk_group, and y
    df = X.copy(deep=True)

    ### START CODE HERE (REPLACE INSTANCES OF 'None' with your code) ###
    # Calculate the baseline risks (use the function that you just implemented)
    baseline_risk = base_risks(df.copy(deep=True), lr)

    # bin patients into 10 risk groups based on their baseline risks
    risk_groups = pd.cut(baseline_risk,10)

    # Store the baseline risk, risk_groups, and y into the new dataframe
    df.loc[:, 'baseline_risk'] = baseline_risk
    df.loc[:, 'risk_group'] = risk_groups
    df.loc[:, 'y'] = y_dev

    # select the subset of patients who did not actually receive treatment
    df_baseline = df[X.TRTMT==False]

    # select the subset of patients who did actually receive treatment
    df_treatment = df[X.TRTMT==True]
```

```
    # For baseline patients, group them by risk group, select their outcome 'y', and take the mean
    baseline_mean_by_risk_group = df_baseline.groupby('risk_group')['y'].mean()

    # For treatment patients, group them by risk group, select their outcome 'y', and take the mean
    treatment_mean_by_risk_group = df_treatment.groupby('risk_group')['y'].mean()

    # Calculate the absolute risk reduction by risk group (baseline minus treatment)
    arr_by_risk_group = baseline_mean_by_risk_group-treatment_mean_by_risk_group

    # Set the index of the arr_by_risk_group dataframe to the average baseline risk of each risk group
    # Use data for all patients to calculate the average baseline risk, grouped by risk group.
    arr_by_risk_group.index = df.groupby('risk_group')['baseline_risk'].mean()

    ### END CODE HERE ###

    # Set the name of the Series to 'ARR'
    arr_by_risk_group.name = 'ARR'


    return arr_by_risk_group
```

In [21]:
```
# Test
abs_risks = lr_ARR_quantile(X_dev, y_dev, lr)

# print the Series
print(abs_risks)

# just showing this as a Dataframe for easier viewing
display(pd.DataFrame(abs_risks))
```

```
baseline_risk
0.202486    0.016667
0.288632   -0.060870
0.358171   -0.059320
0.428335    0.025287
0.509909   -0.063636
0.596385    0.083333
0.670138    0.106719
0.744818    0.072727
0.822216    0.151515
0.912566    0.250000
Name: ARR, dtype: float64
```

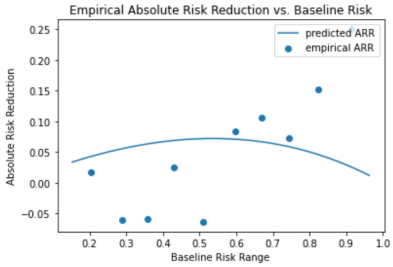
baseline_risk	ARR
0.202486	0.016667
0.288632	-0.060870
0.358171	-0.059320
0.428335	0.025287
0.509909	-0.063636
0.596385	0.083333
0.670138	0.106719
0.744818	0.072727
0.822216	0.151515
0.912566	0.250000

Expected output

```
baseline_risk
0.231595    0.089744
0.314713    0.042857
0.386342   -0.014604
0.458883    0.122222
0.530568    0.142857
0.626937   -0.104072
0.693404    0.150000
0.777353    0.293706
0.836617    0.083333
0.918884    0.200000
Name: ARR, dtype: float64
```

Plot the ARR grouped by baseline risk

In [22]:
```
plt.scatter(abs_risks.index, abs_risks, label='empirical ARR')
plt.title("Empirical Absolute Risk Reduction vs. Baseline Risk")
plt.ylabel("Absolute Risk Reduction")
plt.xlabel("Baseline Risk Range")
ps = np.arange(abs_risks.index[0]-0.05, abs_risks.index[-1]+0.05, 0.01)
diffs = [OR_to_ARR(p, trtmt_OR) for p in ps]
plt.plot(ps, diffs, label='predicted ARR')
plt.legend(loc='upper right')
plt.show()
```



In the plot, the empirical absolute risk reduction is shown as circles, whereas the predicted risk reduction from the logistic regression model is given by the solid line.

If ARR depended only on baseline risk, then if we plotted actual (empirical) ARR grouped by baseline risk, then it would follow the model's predictions closely (the dots would be near the line in most cases).

However, you can see that the empirical absolute risk reduction (shown as circles) does not match the predicted risk reduction from the logistic regression

model (given by the solid line).