PH-291: Exploratory Project

Name: Abhirvey Iyer

Roll.no: 21165003

Supervisor: Dr Ashish Kumar Agrawal

Topic: Predicting Clinical Trial Terminations using ML models

INDEX

Introduction

Clinical trials are studies aiming to determine the validity of an intervention, treatment, or test on human subjects. Randomised controlled trials, where participants are allocated at random (by chance alone) to receive one of several clinical interventions, are the ultimate evaluation of a healthcare intervention. Effective clinical trials are necessary for medical advancements in treating, diagnosing, and understanding diseases. Since 2007, under the Food and Drug Administration Amendments Act (FDAAA), clinical trials are required to be registered to an online database (ClinicalTrials.gov) if they have one or more sites in the United States, conducted under an FDA investigational new drug/devise, or involve a drug/device product manufactured in the U.S. and exported for research. Trials requiring approval of drugs/devices are required to submit results within one year of completion. While the mandate specifies type of trials legally required to submit results, majority of trials with results posted on the database are not legally obligated to do so. The database currently lists 311,260 studies (as of May 2019).

There are many obstacles to conducting a clinical trial. Time frames, number of participants required, and administrative efforts have increased due to several factors:

(1) An industry shift to chronic and degenerative disease research;

(2) Non-novel drug interventions requiring larger trials to identify statistical significance over the existing drug intervention;

(3) Increased complexity of clinical trial protocols;

(4) Increased regulatory barriers.

These factors inflate the financial costs of clinical trials and increase the likelihood of a trial becoming terminated.

Clinical Trials Terminations

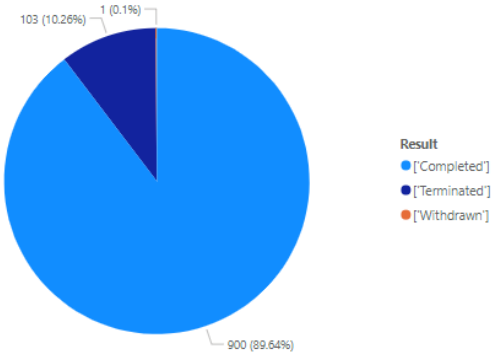
Clinical terminations result in **significant financial burden**. Estimates of drug development are around **1.3 billion dollars** and are rising at a rate of **7.4%,** largely in part to clinical trial costs. Terminated trials are associated with opportunity costs that could have been applied to other efforts. Secondly, there are ethical and scientific issues surrounding terminated clinical trials. All subjects consenting to participate in a clinical trial do so to contribute to the advancement of medical knowledge. If a trial is terminated, subjects are not always informed about the decision and associated reasons, resulting in direct loss of personal benefit from an interventional study. Thirdly, terminated trials also represent a loss of scientific contribution to the community. Often relevant information about why a study was terminated is not reported and results and/or protocols are not published.

A terminated trial indicates that the trial already started recruiting participants but stopped prematurely and recruited participants are no longer being examined/treated. Studies, using **8,000 trials**, found that **10-12%** of clinical trials are terminated. Reasons include insufficient enrolment, scientific data from the trial, safety/efficacy concerns, administrative reasons, external information from a different study, lack of funding/resources and business/sponsor decision.

The Dataset

The dataset used in this report has been derived from a database of 69,000 rows and 640 columns, which in turn has been derived from the site https://clinicaltrials.gov/. This particular dataset used here contains 1005 rows and 640 columns.

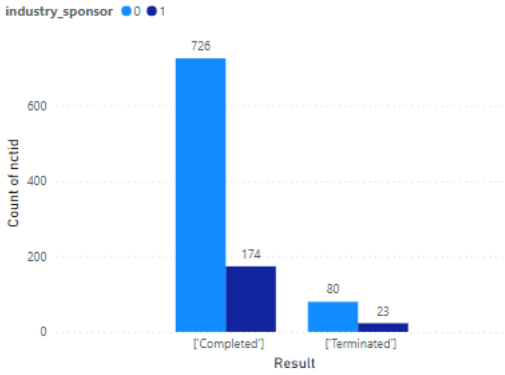
Now, let's do a deep dive analysis of the dataset used here, on which the ML models are trained. This will help us understand the predictions made by the model in a better way.

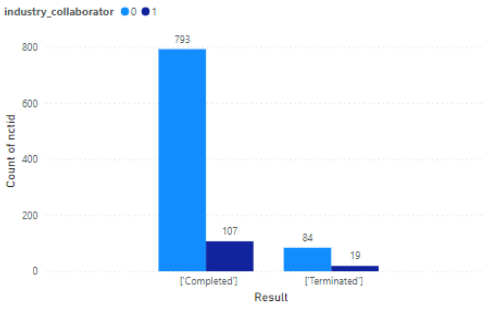


The dataset contains a majority of those clinical trials which have been successfully completed. The distributions as we can see goes this way: **Completed-89.64%, Terminated-10.26%, and Others-0.1%.**

This graph indicates how many completed trials had an industry sponsor and how many didn't, same for the terminated trials.

(0- No industry sponsor, 1- Industry Sponsor present)



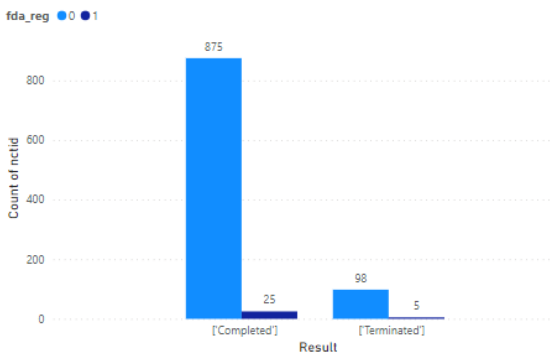


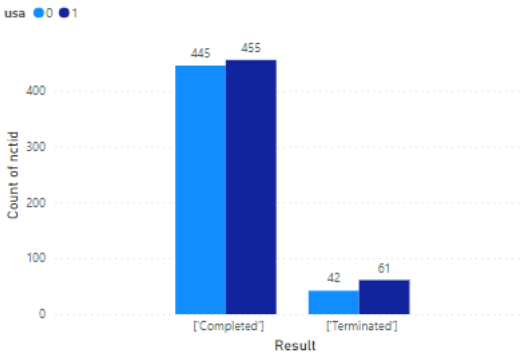
This graph indicates how many completed trials had an industry collaborator and how many didn't, same for the terminated trials.

(0- No industry Collaborator, 1- Industry Collaborator present)

This graph indicates how many trials contained already approved FDA drugs and how many were new drugs, as you can clearly see maximum trials contained new drugs which were previously not in the market as they are not FDA registered

(0- Not registered, 1- Registered)



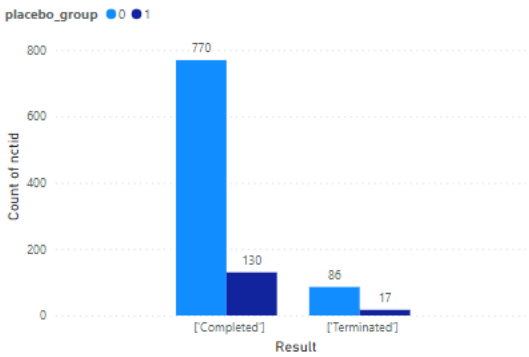


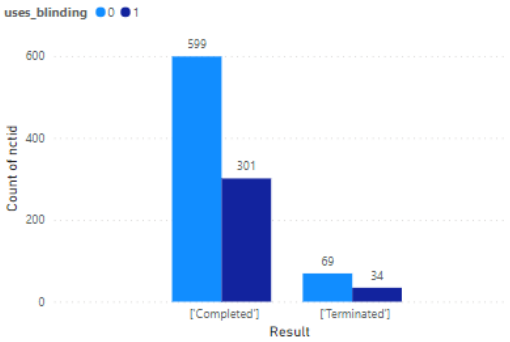
This graph indicates how many completed trials had their main country as USA and how many didn't, same for the terminated trials.

(0- USA is not the main country, 1- USA is the main country)

This graph indicates how many completed trials had a placebo group and how many didn't, same for the terminated trials.

(0- No Placebo, 1- Placebo present)





This graph indicates how many completed trials have used masking and how many didn't, same for the terminated trials.

(0- No masking, 1- Masking used)

ML Models Used

In order to automate the prediction of clinical trials termination, I have used 4 different machine learning algorithms/ models. Those 4 are models are **Logistic Regression, Decision Trees, Random Forest, and XGBoost**. Before moving onto the details of the code let's have a brief look into how do these models work.

- **Logistic Regression:** This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn't vote, based on a given dataset of independent variables.
- **Decision Trees:** A decision tree is a non-parametric supervised learning algorithm, which is utilized for both classification and regression tasks. It has a hierarchical, tree structure, which consists of a root node, branches, internal nodes and leaf nodes.
- **Random Forest:** Random forest is a commonly-used machine learning algorithm trademarked by Leo Breiman and Adele Cutler, which combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.
- **XGBoost:** XGBoost is an optimized distributed gradient boosting library designed for efficient and scalable training of machine learning models. It is an ensemble learning method that combines the predictions of multiple weak models to produce a stronger prediction. XGBoost stands for "Extreme Gradient Boosting" and it has become one of the most popular and widely used machine learning algorithms due to its ability to handle large datasets and its ability to achieve state-of-the-art performance in many machine learning tasks such as classification and regression.

These are the 4 models I have used, so let's finally see how have these models performed in predicting the termination of clinical trials.

Code

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
ed=pd.read_csv("C:\\Users\\rabhi\\OneDrive\\Documents\\Explo_Dataset.csv")
```

In [3]:
```
ed
```

Out[3]:

	Unnamed: 0	nctid	target_label	number_collaborators	number_officials	num_arms	number_countries	number_sites	no_elig_req	inclusion_lines
0	0	NCT01923116	0	2	2	1	1	1	0	6
1	1	NCT01763021	0	0	1	1	1	1	0	2
2	2	NCT01179308	1	0	1	2	2	2	0	3
3	3	NCT00427778	1	1	1	2	1	1	0	3
4	4	NCT00376467	0	0	1	0	1	28	0	3
...	...	...	...	...	...	...	...	...	...	...
1000	1000	NCT01833390	0	3	2	1	1	1	0	0
1001	1001	NCT01953250	0	0	1	0	1	1	0	1
1002	1002	NCT02048046	0	0	0	0	2	7	0	4
1003	1003	NCT01310582	0	1	1	2	1	1	0	4
1004	1004	NCT00738023	0	1	1	2	1	1	0	7

1005 rows × 644 columns

In [4]:
```
ed=ed.drop(['Unnamed: 0'],axis=1)
```

In [5]:
```
ed
```

Out[5]:

	nctid	target_label	number_collaborators	number_officials	num_arms	number_countries	number_sites	no_elig_req	inclusion_lines	exclusion_lir
0	NCT01923116	0	2	2	1	1	1	0	6	
1	NCT01763021	0	0	1	1	1	1	0	2	
2	NCT01179308	1	0	1	2	2	2	0	3	
3	NCT00427778	1	1	1	2	1	1	0	3	
4	NCT00376467	0	0	1	0	1	28	0	3	
...	...	...	...	...	...	...	...	...	...	
1000	NCT01833390	0	3	2	1	1	1	0	0	
1001	NCT01953250	0	0	1	0	1	1	0	1	
1002	NCT02048046	0	0	0	0	2	7	0	4	
1003	NCT01310582	0	1	1	2	1	1	0	4	
1004	NCT00738023	0	1	1	2	1	1	0	7	

1005 rows × 643 columns

In [6]:
```
ed.isnull().sum().sum()
```

Out[6]:
```
0
```

In [7]:
```
ed=ed[ed.Result != 'Error: 500']
```

In [11]:
```
"""The dataset is heavily biased with Completed values so if we use any model we will get results
with accuracy greater than 90%, therefore to get the correct results we need to use sampling techniques"""
```

Out[11]:
```
'The dataset is heavily biased with Completed values so if we use any model we will get results \nwith accuracy greater than 9
0%, therefore to get the correct results we need to use sampling techniques'
```

In [12]:
```
from sklearn.preprocessing import StandardScaler
s=StandardScaler()
```

In [13]:
```
l=ed.select_dtypes(exclude='object').columns
```

In [14]:
```
for f in l:
    ed[f]=s.fit_transform(ed[f].to_numpy().reshape(-1,1))
```

```
In [16]: from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()

In [17]: ed['Result']=le.fit_transform(ed['Result'].astype(str))

In [19]: def correlation(dataset, threshold):
             col_corr = set()  # Set of all the names of correlated columns
             corr_matrix = dataset.corr()
             for i in range(len(corr_matrix.columns)):
                 for j in range(i):
                     if abs(corr_matrix.iloc[i, j]) > threshold: # we are interested in absolute coeff value
                         colname = corr_matrix.columns[i]  # getting the name of column
                         col_corr.add(colname)
             return col_corr

In [20]: x=ed.drop(['Result'],axis=1)
         y=ed['Result']

In [21]: corr_features = correlation(x, 0.7)
         len(set(corr_features))

Out[21]: 39

In [23]: x=x.drop(corr_features,axis=1)

In [25]: x=x.drop(['nctid'],axis=1)

In [27]: from sklearn.model_selection import train_test_split

In [28]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)
```

The above lines code are the pre-requisite steps that are to be applied to the dataset before we apply any type of model to the dataset, as these ensure that the predictions are clean and reduce biases by a significant amount.

The steps applied are:

i) Data Cleaning: It involves removing of all unnecessary rows and columns, and also filling the null values (if any) in the dataset.

ii) Standardization of Numeric Columns: We can standardize the numeric columns of a dataset using various methods, here I have used Standard Scaler technique, which normalises all the data following the distribution where mean is 0 and standard deviation is 1. This makes the work for the ML model easy.

iii) Label Encoding: This involves the conversion of text columns to numeric values.

iv) Feature Selection: Finally, we select the relevant features that affect the output directly and drop all the other columns

v) Splitting the data: Then we split the dataset into train and test datasets, thus making it to ready for any model application.

Now, moving ahead let’s have a look at the performance of all the 4 models mentioned above.

1.) Logistic Regression

In [29]:
```
from sklearn.linear_model import LogisticRegression
```

In [30]:
```
lr=LogisticRegression(class_weight='balanced')
```

In [31]:
```
lr.fit(x_train,y_train)
```

Out[31]:
```
LogisticRegression
LogisticRegression(class_weight='balanced')
```

In [32]:
```
y_pred_lr=lr.predict(x_test)
```

In [33]:
```
y_pred_lr
```

Out[33]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [34]:
```
from sklearn.metrics import classification_report,precision_score,recall_score,f1_score
```

In [35]:
```
precision_score(y_test,y_pred_lr,average='macro')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[35]: 0.6641883519206939

In [36]:
```
precision_score(y_test,y_pred_lr,average='weighted')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[36]: 0.9901154632068737

In [37]:
```
recall_score(y_test,y_pred_lr,average='macro')
```

Out[37]: 0.6568627450980392

In [38]:
```
recall_score(y_test,y_pred_lr,average='weighted')
```

Out[38]: 0.9933774834437086

In [39]:
```
classification_report(y_test,y_pred_lr)
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[39]:
```
'              precision    recall  f1-score   support\n\n           0       0.99      1.00      1.00       267\n           1
1.00      0.97      0.99        34\n           2       0.00      0.00      0.00         1\n\n    accuracy
0.99       302\n   macro avg       0.66      0.66      0.66       302\nweighted avg       0.99      0.99      0.99       302\n'
```

In [40]:
```
from sklearn.metrics import precision_recall_curve
```

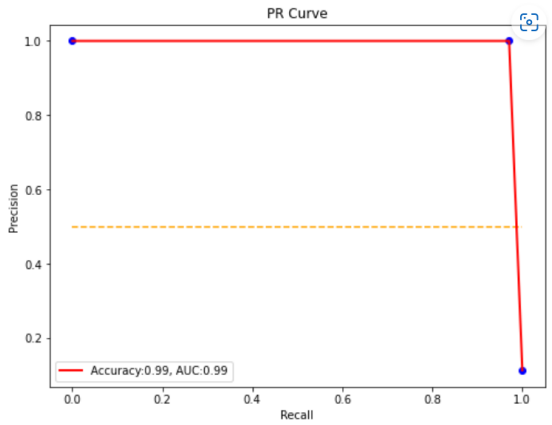
In [41]:
```
precision,recall,thresholds=precision_recall_curve(y_test,y_pred_lr,pos_label=1)
```

In [42]:
```
acc=lr.score(x_test,y_test)
```

In [44]:
```
from sklearn.metrics import auc
p_auc=auc(recall,precision)
```

In [45]:
```
plt.style.context(("ggplot","seaborn"))
plt.figure(figsize=(8,6))
plt.scatter(recall,precision,c='blue')
plt.plot(recall,precision,label="Accuracy:%.2f, AUC:%.2f"%(acc,p_auc),linewidth=2,c='red')
plt.hlines(0.5,0.0,1.0,linestyle='dashed',colors=['orange'])
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("PR Curve")
plt.legend(loc="best")
```

Out[45]: <matplotlib.legend.Legend at 0x211f5f45f40>



The Logistic Regression model gives us a macro average precision of 66.42%, and an area under the PR curve of 0.98.

2.) Decision Trees

In [47]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [48]:
```
dt=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=45,class_weight='balanced')
```

In [49]:
```
dt.fit(x_train,y_train)
```

Out[49]:
```
DecisionTreeClassifier
DecisionTreeClassifier(class_weight='balanced', max_depth=3, random_state=45)
```

In [50]:
```
y_pred_dt=dt.predict(x_test)
```

In [51]:
```
y_pred_dt
```

Out[51]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [52]:
```
precision_score(y_test,y_pred_dt,average='macro')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[52]:
```
0.6641883519206939
```

In [53]:
```
precision_score(y_test,y_pred_dt,average='weighted')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[53]:
```
0.9901154632068737
```

In [54]:
```
recall_score(y_test,y_pred_dt,average='macro')
```

Out[54]:
```
0.6568627450980392
```

In [55]:
```
recall_score(y_test,y_pred_dt,average='weighted')
```

Out[55]:
```
0.9933774834437086
```

In [56]:
```
classification_report(y_test,y_pred_dt)
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[56]:
```
'              precision    recall  f1-score   support\n\n           0       0.99      1.00      1.00       267\n           1
1.00      0.97      0.99        34\n           2       0.00      0.00      0.00         1\n\n    accuracy
0.99       302\n   macro avg       0.66      0.66      0.66       302\nweighted avg       0.99      0.99      0.99       302\n'
```

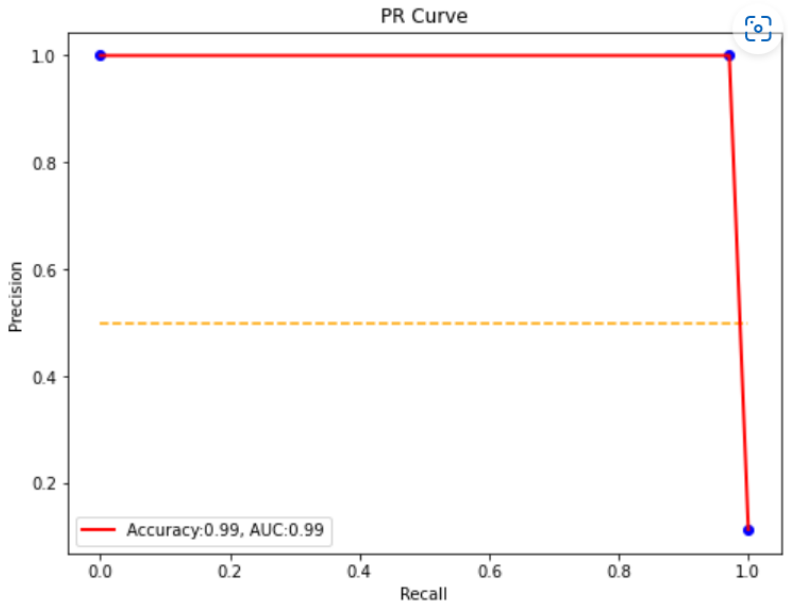
In [57]:
```
precision1,recall1,thresholds1=precision_recall_curve(y_test,y_pred_dt,pos_label=1)
```

In [58]:
```
p_auc1=auc(recall1,precision1)
```

```
In [60]: acc2=dt.score(x_test,y_test)
```

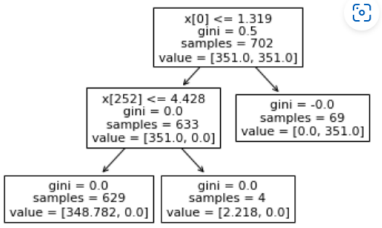
```
In [61]: plt.style.context(("ggplot","seaborn"))
         plt.figure(figsize=(8,6))
         plt.scatter(recall,precision,c='blue')
         plt.plot(recall,precision,label="Accuracy:%.2f, AUC:%.2f"%(acc2,p_auc1),linewidth=2,c='red')
         plt.hlines(0.5,0.0,1.0,linestyle='dashed',colors=['orange'])
         plt.xlabel("Recall")
         plt.ylabel("Precision")
         plt.title("PR Curve")
         plt.legend(loc="best")
```

```
Out[61]: <matplotlib.legend.Legend at 0x211f610a700>
```




```
In [62]: from sklearn import tree
         tree.plot_tree(dt.fit(x_train,y_train))
```

```
Out[62]: [Text(0.6, 0.8333333333333334, 'x[0] <= 1.319\ngini = 0.5\nsamples = 702\nvalue = [351.0, 351.0]'),
          Text(0.4, 0.5, 'x[252] <= 4.428\ngini = 0.0\nsamples = 633\nvalue = [351.0, 0.0]'),
          Text(0.2, 0.16666666666666666, 'gini = 0.0\nsamples = 629\nvalue = [348.782, 0.0]'),
          Text(0.6, 0.16666666666666666, 'gini = 0.0\nsamples = 4\nvalue = [2.218, 0.0]'),
          Text(0.8, 0.5, 'gini = -0.0\nsamples = 69\nvalue = [0.0, 351.0]')]
```



A look of the decision tree used by the model

Decision Trees give us a macro average precision of 66.41% and an area under the PR curve of 0.98.

3.) Random Forest

In [63]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [64]:
```
rf=RandomForestClassifier(n_estimators=10,class_weight='balanced')
```

In [65]:
```
rf.fit(x_train,y_train)
```

Out[65]:
```
RandomForestClassifier
RandomForestClassifier(class_weight='balanced', n_estimators=10)
```

In [66]:
```
y_pred_rf=rf.predict(x_test)
```

In [67]:
```
y_pred_rf
```

Out[67]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [68]:
```
precision_score(y_test,y_pred_rf,average='macro')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[68]:
```
0.2947019867549669
```

In [69]:
```
precision_score(y_test,y_pred_rf,average='weighted')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[69]:
```
0.781643348975922
```

In [70]:
```
recall_score(y_test,y_pred_rf,average='macro')
```

Out[70]:
```
0.3333333333333333
```

In [71]:
```
recall_score(y_test,y_pred_rf,average='weighted')
```

Out[71]:
```
0.8841059602649006
```

In [72]:
```
classification_report(y_test,y_pred_rf)
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
re are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this beha
vior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[72]:
```
'              precision    recall  f1-score   support\n\n           0       0.88      1.00      0.94       267\n           1
0.00      0.00      0.00        34\n           2       0.00      0.00      0.00         1\n\n    accuracy
0.88       302\n   macro avg       0.29      0.33      0.31       302\nweighted avg       0.78      0.88      0.83       302\n'
```

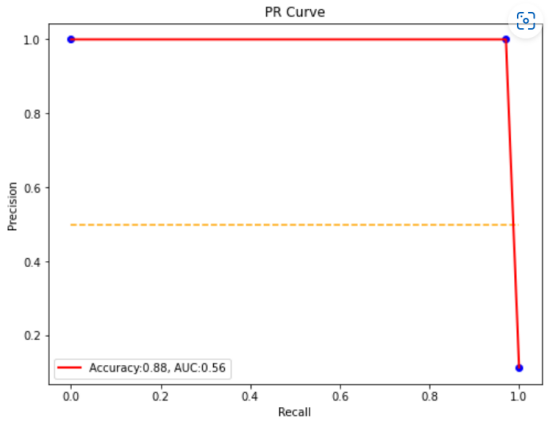
In [73]:
```
precision2,recall2,thresholds2=precision_recall_curve(y_test,y_pred_rf,pos_label=1)
```

In [74]:
```
p_auc2=auc(recall2,precision2)
```

In [76]:
```
acc3=rf.score(x_test,y_test)
```

In [77]:
```
plt.style.context(("ggplot","seaborn"))
plt.figure(figsize=(8,6))
plt.scatter(recall,precision,c='blue')
plt.plot(recall,precision,label="Accuracy:%.2f, AUC:%.2f"%(acc3,p_auc2),linewidth=2,c='red')
plt.hlines(0.5,0.0,1.0,linestyle='dashed',colors=['orange'])
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.title("PR Curve")
plt.legend(loc="best")
```

Out[77]: <matplotlib.legend.Legend at 0x211f6393280>



Random Forest give us a macro average precision of 30% and an area under the PR curve of 0.56.

4.) XGBoost

In [78]:
```
import xgboost as xgb
```

In [79]:
```
smallest_class_count = y_train.sum()
largest_class_count = len(y_train) - smallest_class_count
spw = largest_class_count / smallest_class_count
```

In [80]:
```
xg=xgb.XGBClassifier(scale_pos_weight=spw)
```

In [81]:
```
xg.fit(x_train,y_train)
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is d
eprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=F
alse when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num
_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)

[12:03:58] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'b
inary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

Out[81]:

XGBClassifier

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.300000012,
              max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=8,
              num_parallel_tree=1, predictor='auto', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=9.173913043478262,
              subsample=1, tree_method='exact', validate_parameters=1,
              verbosity=None)
```

In [82]:
```
y_pred_xg=xg.predict(x_test)
```

In [83]:
```
y_pred_xg
```

Out[83]:
```
array([0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [84]:
```
precision_score(y_test,y_pred_xg,average='macro')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[84]:
```
0.6641883519206939
```

In [85]:
```
precision_score(y_test,y_pred_xg,average='weighted')
```

```
C:\Users\rabhi\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning: Precision is ill-de
fined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Out[85]:
```
0.9901154632068737
```

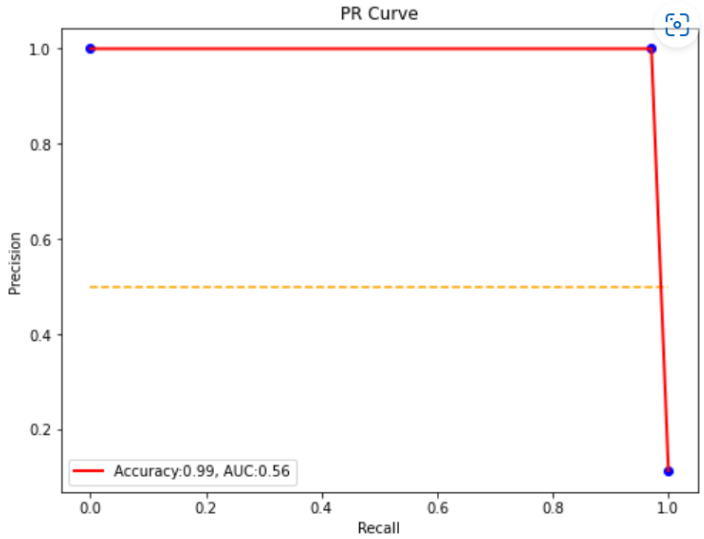
In [86]:
```
recall_score(y_test,y_pred_xg,average='macro')
```

Out[86]:
```
0.6568627450980392
```

In [87]:
```
recall_score(y_test,y_pred_xg,average='weighted')
```

Out[87]:
```
0.9933774834437086
```

```
In [88]: precision3,recall3,thresholds3=precision_recall_curve(y_test,y_pred_rf,pos_label=1)

In [89]: p_auc3=auc(recall3,precision3)

In [91]: acc4=xg.score(x_test,y_test)

In [92]: plt.style.context(("ggplot","seaborn"))
         plt.figure(figsize=(8,6))
         plt.scatter(recall,precision,c='blue')
         plt.plot(recall,precision,label="Accuracy:%.2f, AUC:%.2f"%(acc4,p_auc3),linewidth=2,c='red')
         plt.hlines(0.5,0.0,1.0,linestyle='dashed',colors=['orange'])
         plt.xlabel("Recall")
         plt.ylabel("Precision")
         plt.title("PR Curve")
         plt.legend(loc="best")

Out[92]: <matplotlib.legend.Legend at 0x211f74eea60>
```



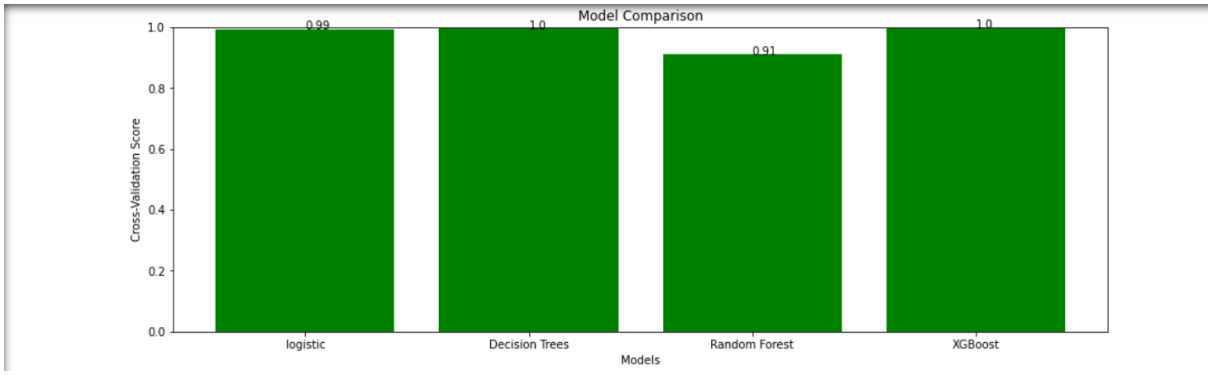
XGBoost give us a macro average precision of 65.68% and an area under the PR curve of 0.56.

As seen above all the 4 models have performed decently well above, let’s compare them now using cross validation so that we can find the right model for accurate prediction at the industry level.

In [93]:
```
from sklearn import model_selection
class Benchmark:
    def plot_cv_results(self):
        plt.figure(figsize=(15,5))
        x = np.arange(len(self.results))
        plt.bar(x, list(self.results.values()), align='center', color ='g')
        plt.xticks(x, list(self.results.keys()))
        plt.ylim([0, 1])
        plt.ylabel('Cross-Validation Score')
        plt.xlabel('Models')
        plt.title('Model Comparison')
        for index, value in enumerate(self.results.values()):
            plt.text(index, value, str(round(value,2)))
        plt.show()


    def __init__(self, models):
        self.models = models

    def test_models(self, X=x, y=y, cv=5):
        self.results = {}
        for name, model in self.models.items():
            scores = model_selection.cross_val_score(model, X, y, cv=cv)
            self.results[name] = scores.mean()
        self.best_model = max(self.results, key=self.results.get)
        return f"The best model is: {self.best_model} with a score of {self.results[self.best_model]:.3f}"
```

In [94]:
```
models = {
    'logistic': lr,
    'Decision Trees': dt,
    'Random Forest': rf ,
    'XGBoost': xg
}

benchmark = Benchmark(models)
print(benchmark.test_models())
benchmark.plot_cv_results()
```

```
The best model is: XGBoost with a score of 0.998
```



According to the cross validation model, XGBoost performs the best with a score of 0.998.

Conclusion

In this project, I used feature engineering and predictive modelling to study key factors associated to clinical trial termination, and proposed a framework to predict trial termination. By using 311,260 clinical trials to build a dataset with 68,999 samples, and taking a sub-sample of 1005 samples, I achieved over **67% Balanced Accuracy** scores for trial termination prediction. The predictive modelling offers insight for stakeholders to better plan clinical trials to avoid waste and ensure success.

A limitation of this approach is that the decision logic of the predictive models is not transparent, making it difficult to interpret the predictions.

PFA:- The entire code.

References

- https://www.nature.com/articles/s41598-021-82840-x#:~:text=By%20using%20sampling%20and%20ensemble,results%20for%20clinical%20trial%20study.
- https://github.com/maggieelkin/ClinicalTrialReports/blob/main/README.md
- https://www.projectpro.io/article/healthcare-machine-learning-projects-with-source-code/508#mcetoc_1firrba3hb
- https://clinicaltrials.gov/ct2/search
- https://blog.bocabio.com/accurately-predict-clinical-trial-terminations