Data Glacier

Your Deep Learning Partner

Healthcare - Drug Persistence

Groupe name: Ehiafarin
Country: Iran
Groupe member: Alireza Ehiaei
Email: arh.ehiaei@yahoo.com

Specialization: Data Science
Final Report
September-2021

Table Of Contents

Introduction

In this document, a comprehensive report of research conducted in previous practices is provided. It includes problem statement, data understanding, cleaning data, model development, etc.

In order to more clarify the terminologies of the problem and methods to approach it, before each part some useful definitions have been provided.

Also, the complete notebook code has been attached to the GitHub link mentioned in the last slide.

Report lifecycle

- Understanding the problem
- Modifying and Cleaning data
- Introduction of machine learning models
- Developing classification models
- Evaluating developed models in different ways
- Report the accuracy of the model

Problem Statement

Measuring patient persistency with drug therapy provides valuable information for healthcare decision makers concerning the effectiveness of a drug in a routine practice setting, which epidemiologists call the population based setting, as opposed to the trial- or clinic-based setting.

One of the challenge for all Pharmaceutical companies is to understand the persistency of drug as per the physician prescription. To solve this problem ABC pharma company approached an analytics company to automate this process of identification.

Business understanding

Increasing adherence rates by only 10 percentage points would translate into a $41 billion pharmaceutical revenue opportunity in the US ($124 billion globally), accompanied by improved health outcomes and decreased healthcare spending. [1]

1 _ https://medipense.com/medication-adherence-compliance/

Glossary

Persistency Definitions [1]

The measurement of medication persistency attempts to capture the amount of time that an individual remains on chronic drug therapy. Under this framework, patients are classified as either persistent or non persistent with medication therapy for some duration of time. Individuals who are persistent with therapy are continuous with their medication-taking behavior during a certain period. Persistent individuals refill their medications frequently and regularly. In contrast, no persistent individuals either have sporadic refilling practices or have discontinued refilling their medications completely.

1 -https://www.ajmc.com/view/jul05-2085p449-457

Glossary

Medication adherence vs medication persistence [1]

Medication adherence (compliance): “the extent to which a patient acts in accordance with the prescribed interval and dose of a dosing regimen.”

Medication persistence: “the duration of time from initiation to discontinuation of therapy.”

1. https://www.pharmacytoday.org/article/S1042-0991(15)30340-6/fulltext#relatedArticles

Data understanding

There are 3424 rows (patients) and 69 coloumns (features) in which except columns ‘Dexa Freq During Rx’ and ‘Count Of Risks’ which are integer, other coloumns have string type.

```
1 drug_Data.head()
```

	Ptid	Persistency_Flag	Gender	Race
0	P1	Persistent	Male	Caucasian
1	P2	Non-Persistent	Male	Asian
2	P3	Non-Persistent	Female	Other/Unknown
3	P4	Non-Persistent	Female	Caucasian
4	P5	Non-Persistent	Female	Caucasian

```
1 drug_Data.shape
```

```
(3424, 69)
```

```
1 type_dct = {str(k): list(v) for k, v in df.groupby(df.dtypes, axis=1)}
```

```
1 type_dct
```

```
{'int64': ['Dexa_Freq_During_Rx', 'Count_Of_Risks'],
 'object': ['Ptid',
  'Persistency_Flag',
  'Gender',
  'Race',
  'Ethnicity',
  'Region',
  'Age_Bucket',
  'Ntm_Speciality',
  'Ntm_Specialist_Flag',
  'Ntm_Speciality_Bucket',
  'Gluco_Record_Prior_Ntm',
  'Gluco_Record_During_Rx',
  'Dexa_During_Rx',
```

Data understanding

From 3424 cases, there are 1289 drug persistent patients and 2135 non persistent patients.

```
1 df_Persistent = df[df['Persistency_Flag'] == 'Persistent']
2 df_Non_Persistent = df[df['Persistency_Flag'] == 'Non-Persistent']
```

```
1 len(df_Persistent.iloc[:,1])
```

```
1289
```

```
1 len(df_Non_Persistent.iloc[:,1])
```

```
2135
```

Data understanding

From 3424 cases, there are just 194 man and 3230 women showing a significant difference in gender factor on drug persistency.

```
1  df_Male = df[df['Gender'] == 'Male']
2  df_Female = df[df['Gender'] == 'Female']
```

```
1  len(df_Male)
```

```
194
```

```
1  len(df_Female)
```

```
3230
```

Feature Description

Bucket (variable)

Variable Description

Unique Row Id (patient ID)

Unique ID of each patient

Target Variable (persistency Flag)

Flag indicating if a patient was persistent or not

Age

Age of the patient during their therapy

Race

Race of the patient from the patient table

Region

Region of the patient from the patient table

Feature Description

Demographics

Ethnicity:

Ethnicity of the patient from the patient table

Gender :

Gender of the patient from the patient table

IDN Indicator:

Flag indicating patients mapped to IDN

Feature Description

Provider Attributes

NTM - Physician Specialty

Specialty of the HCP that prescribed the NTM Rx

NTM - T-Score

T Score of the patient at the time of the NTM Rx (within 2 years prior from rxdate)

Change in T Score

Change in Tscore before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown)

NTM - Risk Segment

Risk Segment of the patient at the time of the NTM Rx (within 2 years days prior from rxdate)

Change in Risk Segment

Change in Risk Segment before starting with any therapy and after receiving therapy (Worsened, Remained Same, Improved, Unknown)

NTM - Multiple Risk Factors

Flag indicating if patient falls under multiple risk category (having more than 1 risk) at the time of the NTM Rx (within 365 days prior from rxdate)

Feature Description

Provider Attributes

NTM - Dexa Scan Frequency

Number of DEXA scans taken prior to the first NTM Rx date (within 365 days prior from rxdate)

NTM - Dexa Scan Recency

Flag indicating the presence of Dexa Scan before the NTM Rx (within 2 years prior from rxdate or between their first Rx and Switched Rx; whichever is smaller and applicable**)**

Dexa During Therapy

Flag indicating if the patient had a Dexa Scan during their first continuous therapy

NTM - Fragility Fracture Recency

Flag indicating if the patient had a recent fragility fracture (within 365 days prior from rxdate)

Fragility Fracture During Therapy

Flag indicating if the patient had fragility fracture during their first continuous therapy

Feature Description

Provider Attributes

NTM - Glucocorticoid Recency

Flag indicating usage of Glucocorticoids (>=7.5mg strength) in the one year look-back from the first NTM Rx

Glucocorticoid Usage During Therapy

Flag indicating if the patient had a Glucocorticoid usage during the first continuous therapy

NTM - Injectable Experience

Flag indicating any injectable drug usage in the recent 12 months before the NTM OP Rx

NTM - Risk Factors

Risk Factors that the patient is falling into. For chronic Risk Factors complete lookback to be applied and for non-chronic Risk Factors, one year lookback from the date of first OP Rx

Feature Description

Disease/Treatment Factor:

NTM - Comorbidity

Comorbidities are divided into two main categories - Acute and chronic, based on the ICD codes. For chronic disease we are taking complete look back from the first Rx date of NTM therapy and for acute diseases, time period before the NTM OP Rx with one year lookback has been applied

NTM - Concomitancy

Concomitant drugs recorded prior to starting with a therapy(within 365 days prior from first rxdate)

Adherence

Adherence for the therapies

Data problem

As it is shown in table below, there is no non value in data, but…

```
:  1  def missing_values_table(df):
   2      mis_val = df.isnull().sum()
   3      mis_val_percent = 100 * df.isnull().sum() / len(df)
   4      mis_val_table = pd.concat([mis_val, mis_val_percent], axis=1)
   5      mis_val_table_ren_columns = mis_val_table.rename(
   6      columns = {0 : 'Missing Values', 1 : '% of Total Values'})
   7      mis_val_table_ren_columns = mis_val_table_ren_columns[
   8          mis_val_table_ren_columns.iloc[:,1] != 0].sort_values(
   9      '% of Total Values', ascending=False).round(1)
  10      print ("Your selected dataframe has " + str(df.shape[1]) + " columns.\n"
  11          "There are " + str(mis_val_table_ren_columns.shape[0]) +
  12              " columns that have missing values.")
  13      return mis_val_table_ren_columns
```

```
:  1  missing_values_table(df)
```

```
Your selected dataframe has 69 columns.
There are 0 columns that have missing values.
```

:

Missing Values	% of Total Values

Data problem

Supposedly, there is no nan data, but there are some unknown values? For example, ‘Risk segment during rx’ column has 1497 unknown values. We treat with them as missing values in different ways.

```
new_df = df.Risk_Segment_During_Rx.str.split(expand=True).stack().value_counts().reset_index()

new_df.columns = ['Word', 'Frequency']

new_df
```

	Word	Frequency
0	Unknown	1497
1	HR_VHR	965
2	VLR_LR	962

Data problem

Threating with unknown data

In order to treat with unknown values we can use techniques such as:

- Exclude unknown values. if the remaining data set is not imbalanced.
- Replace / group the unknown values with an appropriate value - e.g. replace missing values with the most populous label in the column.
- Targeting the column including unknown values and forecasting the label that each row (patient) likely should has instead of unknown value.

Problem Statement

Delete coloumns or rows with missing values:

Missing values can be handled by deleting the rows or columns having null values. For example, one can choose If 50% or more of the rows as null then the entire column can be dropped. Similarly, If a row has e.g. one or more columns values as null can also be dropped.

Dealing with missing values, method 1

Dropping columns or rows including missing values:

For example, we can remove the columns with more than 30%, and drop the rows with 20% missing values. There are 3424 rows, we remove 4 columns which have more than 30% missing values.

```
1  list_column_with_nulls = []
2
3  for column in df.columns:
4      n_missed = len(df[df.loc[:,column]==
5      percent = n_missed/3424
6      if percent > 0.3:
7          print(column)
8          print("percentage of missing valu
9          list_column_with_nulls.append(col
```

```
Risk_Segment_During_Rx
percentage of missing values : 0.44
Tscore_Bucket_During_Rx
percentage of missing values : 0.44
Change_T_Score
percentage of missing values : 0.44
Change_Risk_Segment
percentage of missing values : 0.65
```

```
1  len(list_column_with_nulls)
```

```
4
```

Dealing with missing values

New data set has 4 less columns.

```
1 drug_Data.head()
```

	Ptid	Persistency_Flag	Gender	Race	Ethnicity	Regi
0	P1	Persistent	Male	Caucasian	Not Hispanic	We
1	P2	Non-Persistent	Male	Asian	Not Hispanic	We
2	P3	Non-Persistent	Female	Other/Unknown	Hispanic	Midwe
3	P4	Non-Persistent	Female	Caucasian	Not Hispanic	Midwe
4	P5	Non-Persistent	Female	Caucasian	Not Hispanic	Midwe

```
1 drug_Data.shape
```

```
(3424, 69)
```

```
1 df = drug_Data
```

```
1 df.drop([col for col in list_colum
```

```
1 df.head()
```

	Ptid	Persistency_Flag	Gender	Ra
0	P1	Persistent	Male	Caucasi
1	P2	Non-Persistent	Male	Asi
2	P3	Non-Persistent	Female	Other/Unknov
3	P4	Non-Persistent	Female	Caucasi
4	P5	Non-Persistent	Female	Caucasi

```
1 df.shape
```

```
(3424, 65)
```

Dealing with missing values

Similarly, we drop the rows with just 2% missing values.

```
rows_with_missing_values = []
for i in range(len(df)) :
    n=0
    s = []
    for j in df.columns:
        if (df.loc[i,j]=='Other/Unknown') | (df.loc[i,j]=='Unknown'):
            n=n+1
            s.append(j)
    if (n/69 > 0.02):
        print('row :',i, '    number of missing values :', n)
        print('Columns including missing values :',s)
        rows_with_missing_values.append(i)
```

```
row : 154     number of missing values : 2
Columns including missing values : ['Race', 'Ethnicity']
row : 218     number of missing values : 2
Columns including missing values : ['Race', 'Ethnicity']
row : 221     number of missing values : 2
Columns including missing values : ['Race', 'Ntm_Speciality']
row : 406     number of missing values : 2
Columns including missing values : ['Race', 'Ethnicity']
```

Dealing with missing values

Finally, we have a data set with 3379 rows and 65 columns.

	Ptid	Persistency_Flag	Gender	Rac
0	P1	Persistent	Male	Caucasia
1	P2	Non-Persistent	Male	Asia
2	P3	Non-Persistent	Female	Other/Unknow
3	P4	Non-Persistent	Female	Caucasia
4	P5	Non-Persistent	Female	Caucasia

```
1 df.shape
```

```
(3379, 65)
```

Dealing with missing values, method 2

Imputation

Since the columns including missing values in this research are categorical, their missing values can be replace with the most frequent category. We could use this method at first step, but If the number of missing values is very large the accuracy of final prediction would be reduced. Now, we removed some rows and columns, and impute the rest of missing values in each column with the mode of that column.

Dealing with missing values

By assuming ‘other/unknown’ and ‘unknown’ values as null, still there are four columns including missing values:

```
1 n_null = df.isnull().sum(axis = 0)
2 n_null.head(60)
```

```
Gender                    0
Race                     64
Ethnicity                58
Region                   55
Age_Bucket                0
Ntm_Speciality          290
Ntm Specialist Flag       0
```

Dealing with missing values

After replacing nulls with the mode of each column, no missing value remains.

Filling nulls of each column with the most frequent (mode) of each column

```
1 df = df.fillna(df.mode().iloc[0])
```

Any null in data?

```
1 df.isnull().values.any()
```

```
False
```

Dealing with missing values, method 3

Prediction of missed values

Another way to treat with missing values is predicting them. For example, assume that we want to predict all null values of the column ‘Risk_Segment_During_Rx’ based on other columns. First, object data type have been converted to float type to be used in the LinearDiscriminantAnalysis model. Then, by finding missed values in ‘Risk_Segment_During_Rx’ column and removing their rows we have data set for trainning the model.

Dealing with missing values

Finally, by using the model the missed values of the last column are predicted and filled.

```
from numpy import nan
from pandas import read_csv
from sklearn.discriminant_analysis import LinearDiscriminantAnalysi
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

X = Data_without_nulls.iloc[:,:-1]
y = Data_without_nulls.iloc[:,-1]
# define the model
model = LinearDiscriminantAnalysis()
# define the model evaluation procedure
cv = KFold(n_splits=3, shuffle=True, random_state=1)
# evaluate the model
result = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
# report the mean performance
print('Accuracy: %.3f' % result.mean())
```

```
Accuracy: 0.926
```

Dealing with missing values

Finally, there is no missed values:

Predicting the last column using the model

```
]:  1  data_ =[]
    2  data_ =pd.DataFrame(data_)
    3  data_ = pd.DataFrame(model.predict(data_without_null_1.iloc[:,:-1]))
    4
```

Replacing predicted values just for missed values

```
]:  1  for i in range(len(data_without_null_1)) :
    2      if(pd.isnull(data_without_null_1.loc[i,'Risk_Segment_During_Rx'])):
    3            data_without_null_1.loc[i,'Risk_Segment_During_Rx']= data_.iloc[i,0]
```

Problem Solving Approach

Due to the high number of features, It is hard to identify main factors causing the drug persistence. Drug discovery process generally, and drug persistency specifically are multifaceted problems that in many cases they seem irrelevant and traditional methods. For example, an initial analysis shows the risk factor of region, has low effect on other features, as well drug persistence. But it is not a useful method analyze relationship between different features separately.

```
1 risk_region = df.groupby(['Region']).agg({'Count_Of_Risks':'mean'}).reset_index()
2 risk_region
```

	Region	Count_Of_Risks
0	Midwest	1.10
1	Northeast	1.32
2	Other/Unknown	1.17
3	South	1.41
4	West	1.18

Problem Solving Approach

Since, there is little empirical evidence that the traditional methods have high accuracy in such a this problem, prediction of drug persistency of a patient based on different categorical variables, machine learning methods would be best approach to solve the problem.

After dealing with missing data, all columns except the persistence one are assigned to a data frame called X as input variables, and the column persistence is added to the data frame as the y variable. Now, we have an introduction to machine learning and the ML methods used in this research.

Machine learning models

Machine learning [1]

In general, a learning problem considers a set of n samples of data and then tries to predict properties of unknown data. This problem can be either:

1. supervised learning, in which the data comes with additional attributes that the goal is predicting them, including the prediction models of:

- **classification**: samples belong two or more classes and we want to learn from already labeled data how to predict the class of unlabeled data.
- **regression**: if the desired output consists of one or more continuous variables, then the task is called regression.

1 - https://scikit-learn.org/stable/tutorial/basic/tutorial.html

Machine learning models

Machine learning

2. unsupervised learning, in which the training data consists of a set of input vectors x without any corresponding target values and the goal in such problems may be:

- **Clustering**: discovering groups of similar examples within the data.
- **Density estimation**: determining the distribution of data within the input space, or to project the data from a high-dimensional space down to two or three dimensions for the purpose of visualization.

Machine learning models

Therefore, supervised Learning is applied when we have a labelled data set (the output variable/dependent variable). For example, a data set which contains the size of the house (independent variable) and corresponding house price (dependent variable). We can predict the house price of new data points with respect to the size of the house. Another example is, determining if a tumor is harmful or not harmful when we already have a list of tumors which are harmful or not. In supervised learning we know the problem statement and have all the necessary features to get answer.

Machine learning models

But in unsupervised Learning, we do not have labelled data. We do not have any output variable. We do not know the problem statement. It is applied when we need to find a structure in the data set and extract meaningful insights out of it. For example, a data set of Walmart containing its customer's buying pattern.

Classification and regression algorithms are used when dealing with a supervised learning problem and clustering algorithms are used when dealing with unsupervised learning.

Problem Statement & Machine learning

Drug persistency detection can be identified as a classification problem, this is a binary classification since there can be only two classes i.e has persistency or does not have persistency. The classifier, in this case, needs training data to understand how the given input variables are related to the class. And once the classifier is trained accurately, it can be used to detect whether persistency is there or not for a particular patient.

Classification models

Since the problem in this research is a classification problem, a short definition about classification is necessary.

Classification is a process of categorizing a given set of data into classes, It can be performed on both structured or unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

The classification predictive modeling is the task of approximating the mapping function from input variables to discrete output variables. The main goal is to identify which class/category the new data will fall into.

Classification models

Three major classification of ML algorithms are:

1) Classification Algorithms - Naive Bayes Classification, Decision Tree, Random Forest, kNN, Support Vector Machine (SVM), Neural Networks, etc.

2) Regression Algorithms - Linear Regression, Logistic Regression, Lasso Regression, etc. (Note: Although Logistic Regression has Regression in its name, it is essentially a classification algorithm).

3) Clustering Algorithms - K-Means Clustering, Fuzzy C Means, Mixture of Gaussian, etc.

Classification Terminologies In Machine Learning

Classifier – It is an algorithm that is used to map the input data to a specific category.

Classification Model – The model predicts or draws a conclusion to the input data given for training, it will predict the class or category for the data.

Feature – A feature is an individual measurable property of the phenomenon being observed.

Binary Classification – It is a type of classification with two outcomes, for eg – either true or false.

Multi-Class Classification – The classification with more than two classes, in multi-class classification each sample is assigned to one and only one label or target.

Multi-label Classification – This is a type of classification where each sample is assigned to a set of labels or targets.

Initialize – It is to assign the classifier to be used for the

Train the Classifier – Each classifier in sci-kit learn uses the fit(X, y) method to fit the model for training the train X and train label y.

Predict the Target – For an unlabeled observation X, the predict(X) method returns predicted label y.

Evaluate – This basically means the evaluation of the model i.e classification report, accuracy score, etc.

Data insight

The features in this research are categorical variables. These variables are typically stored as text values which represent various traits. Some examples include gender, age_bucket, risk_low_calcium_intake, risk_vitamin_D_insufficiency. Regardless of what the value is used for, the challenge is determining how to use this data in machine learning. Many machine learning algorithms can support categorical values without further manipulation but there are many more algorithms that do not. Therefore, we converted data type to numbers to be able to use them in different methods of machine learning.

Machine learning models

In this research, different supervised machine learning algorithm from Scikit-learn is used to create a model predicting the targeted variable. Scikit-learn is a free software machine learning library for the Python programming language.

We used three different classification methods explained below to develop predicting models and fitted the models across X and y.

Finally, the model can be used in order to predict whether a patient given a set of information of features in X, is persistent in drug administration or not?

Random Forest Models

Random decision trees or random forest operate by constructing a multitude of decision trees at training time and outputs the class that is the mode of the classes or classification or mean prediction(regression) of the individual trees.

A random forest is a meta-estimator that fits a number of trees on various subsamples of data sets and then uses an average to improve the accuracy in the model's predictive nature. The sub-sample size is always the same as that of the original input size but the samples are often drawn with replacements.

Random Forest Models

Advantages and Disadvantages

The advantage of the random forest is that it is more accurate than the decision trees due to the reduction in the over-fitting. The only disadvantage with the random forest classifiers is that it is quite complex in implementation and gets pretty slow in real-time prediction.

Random Forests Models

Since random is used widely in Classification and Regression problems, we used it as the first method. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

```
1  import sklearn as sk
2  from sklearn.ensemble import RandomForestClassifier
3
4  y = data_without_null_1.iloc[:,1]
5  X = data_without_null_1.iloc[:,1:]
6
7  RF = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
8
```

Classifier Model Evaluation

Any prediction models need to be evaluated to check its accuracy and efficiency. There are a lot of ways to evaluate a classifier model. For example, k-Fold Cross-Validation and ROC Curve that we used in addition to

In k-Fold Cross-Validation a total of k models are fit and evaluated on the k hold-out test sets and the mean performance is reported. In receiver operating characteristics or ROC curve is used for visual comparison of classification models, which shows the relationship between the true positive rate and the false positive rate. The area under the ROC curve is the measure of the accuracy of the model.

But one can use other methods of evaluating the accuracy of machine learning like classification report and the holdout method.

k-Fold Cross-Validation

In the above code k-Fold Cross-Validation is used to determine the accuracy of the model (0.89). It is common to evaluate machine learning models on a dataset using k-fold cross-validation.

The k-fold cross-validation procedure divides a limited dataset into k non-overlapping folds. Each of the k folds is given an opportunity to be used as a held-back test set, whilst all other folds collectively are used as a training dataset. A total of k models are fit and evaluated on the k hold-out test sets and the mean performance is reported.

Random Forests Models

We can again fit them using sklearn, and use them to predict outcomes, as well as get mean prediction accuracy. The accuracy of this method on our data is 0.89.

```
RF.fit(X, y)

# define the model evaluation procedure
cv = KFold(n_splits=3, shuffle=True, random_state=1)
# evaluate the model
result = cross_val_score(RF, X, y, cv=cv, scoring='accuracy')
# report the mean performance
print('Accuracy of the model is: %.3f' % result.mean())
```

```
Accuracy of the model is: 0.897
```

ROC Curves

In this part, we discover Receiver Operating Characteristic curve, or ROC Curves that is used to interpret the prediction of probabilities for binary classification problems.

When making a prediction for a binary or two-class classification problem, there are two types of errors that we could make.

False Positive. Predict an event when there was no event.

False Negative. Predict no event when in fact there was an event.

ROC Curves

ROC is a plot of the false positive rate (x-axis) versus the true positive rate (y-axis) for a number of different candidate threshold values between 0.0 and 1.0. Put another way, it plots the false alarm rate versus the hit rate.

The true positive rate is calculated as the number of true positives divided by the sum of the number of true positives and the number of false negatives. It describes how good the model is at predicting the positive class when the actual outcome is positive.

ROC Curves summarize the trade-off between the true positive rate and false positive rate for a predictive model using different probability thresholds.

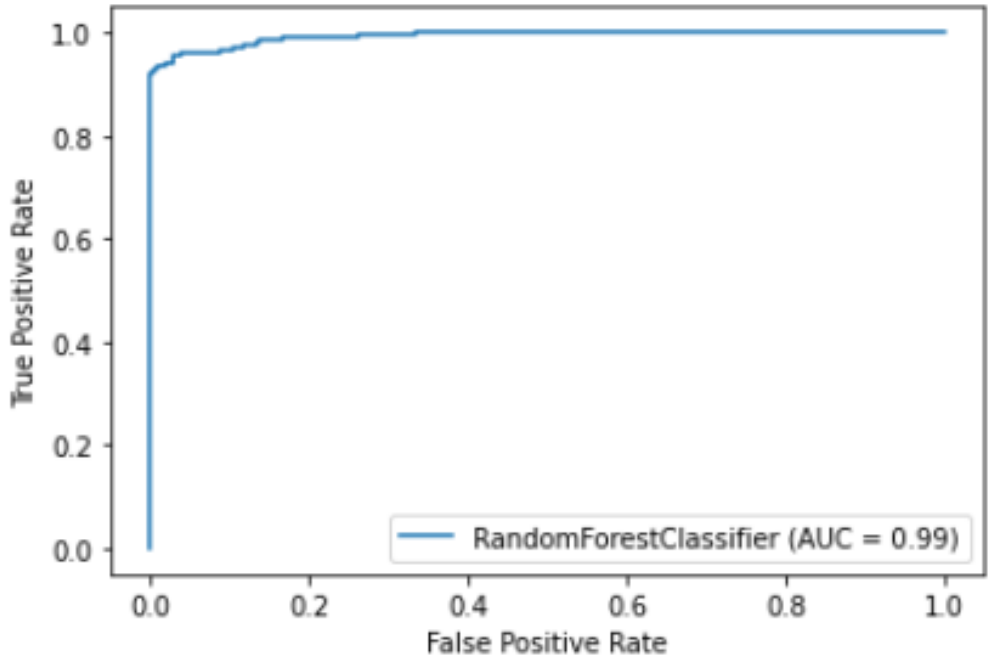
ROC Curves

The area under the curve (AUC) can be used as a summary of the model skill. A skilful model will assign a higher probability to a randomly chosen real positive occurrence than a negative occurrence on average. This is what we mean when we say that the model has skill. Generally, skilful models are represented by curves that bow up to the top left of the plot.

A no-skill classifier is one that cannot discriminate between the classes and would predict a random class or a constant class in all cases.

ROC Curves

As it is shown the are under curve for the random forest we developed is big enough to trust to the predictions of the model.



Prediction

Now, we can use the developed model to predict the drug persistency of a patient. For example, assume we want to use it for a patient with information mentioned in the row 3000:

```
yhat = RF.predict([list(data_without_null_1.iloc[3000,1:])])
if yhat == 1:
    print('The patient is predicted to be persistent' )
else:
    print('The patient is predicted to be non-persistent')
```

```
The patient is predicted to be persistent
```

Logistic Regression Model

It is a classification algorithm in machine learning that uses one or more independent variables to determine an outcome. The outcome is measured with a dichotomous variable meaning it will have only two possible outcomes.

The goal of logistic regression is to find a best-fitting relationship between the dependent variable and a set of independent variables. It is better than other binary classification algorithms like nearest neighbor since it quantitatively explains the factors leading to classification.

Logistic Regression Model

Logistic Regression is a type of Generalized Linear Model (GLM) that uses a logistic function to model a binary variable based on any kind of independent variable.

Logistic regression is used to obtain an odds ratio in the presence of more than one explanatory variable. The procedure is quite similar to multiple linear regression, with the exception that the response variable is binomial.

Logistic Regression Model

Advantages and Disadvantages

Logistic regression is specifically meant for classification, it is useful in understanding how a set of independent variables affect the outcome of the dependent variable.

The main disadvantage of the logistic regression algorithm is that it only works when the predicted variable is binary, it assumes that the data is free of missing values and assumes that the predictors are independent of each other.

Logistic Regression Model

Logistic regression is applied to predict the categorical dependent variable, and our problem is predicting a binary target variable. So, logistic regression can be a good choice to solve our classification.

```
import sklearn as sk
from sklearn.linear_model import LogisticRegression
import pandas as pd
import os

y = data_without_null_1.iloc[:,1]
X = data_without_null_1.iloc[:,1:]

LR = LogisticRegression(random_state=0).fit(X, y)
# define the model evaluation procedure
cv = KFold(n_splits=3, shuffle=True, random_state=1)
# evaluate the model
result = cross_val_score(model, X, y, cv=cv, scoring='accuracy')
# report the mean performance
print('Accuracy: %.3f' % result.mean())
```

```
Accuracy: 0.811
```

Neural Networks Models

The last algorithm used to create prediction model is neural network . A neural network consists of neurons that are arranged in layers, they take some input vector and convert it into an output. The process involves each neuron taking input and applying a function which is often a non-linear function to it and then passes the output to the next layer.

In general, the network is supposed to be feed-forward meaning that the unit or neuron feeds the output to the next layer but there is no involvement of any feedback to the previous layer.

Weighings are applied to the signals passing from one layer to the other, and these are the weighings that are tuned in the training phase to adapt a neural network for any problem statement.

Neural Networks Models

Neural Networks are a machine learning algorithm that involves fitting many hidden layers used to represent neurons that are connected with synaptic activation functions. These essentially use a very simplified model of the brain to model and predict data.

They consists of an artificial network of functions, called parameters, which allows the computer to learn, and to fine tune itself, by analyzing new data. Each parameter (neurons) is a function which produces an output, after receiving one or multiple inputs and then passed to the next layer of neurons several times and produce further outputs until every layer of neurons have been considered, and the terminal neurons have received their input. Those terminal neurons then output the final result for the model.

Neural Networks Models

Advantages and Disadvantages

It has a high tolerance to noisy data and able to classify untrained patterns, it performs better with continuous-valued inputs and outputs. The disadvantage with the artificial neural networks is that it has poor interpretation compared to other models.

Neural Networks models

Neural Networks are well known techniques for classification problems. They can also be applied to regression problems.

Neural Networks

```
:
```

```
1
2  import sklearn as sk
3  from sklearn.neural_network import MLPClassifier
4
5  y = data_without_null_1.iloc[:,1]
6  X = data_without_null_1.iloc[:,1:]
7
8  NN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
9  NN.fit(X, y)
10
11 # define the model evaluation procedure
12 cv = KFold(n_splits=3, shuffle=True, random_state=9)
13 # evaluate the model
14 result = cross_val_score(NN, X, y, cv=cv, scoring='accuracy')
15 # report the mean performance
16 print('Accuracy: %.3f' % result.mean())
```

```
Accuracy: 0.979
```

Prediction

As it is shown, for the patient number 3000 the prediction is not to be drug persistence. The question arises that which model is better?

```
1  yhat = NN.predict([list(data_without_null_1.iloc[3000,1:])])
2  if yhat == 1:
3      print('The patient is predicted to not have drug persistence.' )
4  else:
5      print('The patient is predicted to not have drug persistence.')
```

```
The patient is predicted to not have drug persistence.
```

Ensemble Modeling

An ensemble is a supervised learning technique for combining multiple weak learners/ models to produce a strong learner .

It is possible to combine multiple models of same ML algorithms, but combining multiple predictions generated by different algorithms would normally lead to better predictions.

For example, the predictions of a random forest, a KNN, and a Naive Bayes may be combined to create a stronger final prediction set as compared to combining three random forest model.

Ensemble Modeling

Therefore, the key to creating a powerful ensemble is model diversity leading to two major benefits of Ensemble models: better prediction and more stable model.

We have three ML models, random forest, logistic regression and neural networks which belong to different groups of classification and regression algorithms. So, they are diverse enough in nature that lead to increase performance of ensemble model we are going to develop using them.

Ensemble Techniques; Max Voting

The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point.

The predictions by each model are considered as a ‘vote’. The predictions which we get from the majority of the models are used as the final prediction.

```
from sklearn.ensemble import VotingClassifier

model = VotingClassifier(estimators=[('RF', RF), ('NN', NN), ('LR', LR)], voting='hard')
model.fit(X, y)

```

Ensemble Techniques; Max Voting

Finally, the ensemble model is ready to predict the target variable.

```
7  yhat_model = model.predict([list(data_without_null_1.iloc[3000,1:])])
8  if yhat_model == 1:
9      print('The patient is predicted to not have drug persistence.' )
10 else:
11     print('The patient is predicted to not have drug persistence.')
```

```
The patient is predicted to not have drug persistence.
```

Recommendation to technical users

Others machine learning methods can be recommended to professional users. For example, support vector machine.

The support vector machine is a classifier that represents the training data as points in space separated into categories by a gap as wide as possible. New points are then added to space by predicting which category they fall into and which space they will belong to. It uses a subset of training points in the decision function which makes it memory efficient and is highly effective in high dimensional spaces. The only disadvantage with the support vector machine is that the algorithm does not directly provide probability estimates.

Thank You