Data Science Intern at Data Glacier

Project: Healthcare - Persistency of a drug

Week 13: Final report

Name: Amraoui Fatima Ezzahra

Email: amraouifatimaezzahra@gmail.com

Country: Morocco

Specialization: Data Science

Batch Code: LISUM13

Date: 30 November 2022

Submitted to: Data Glacier

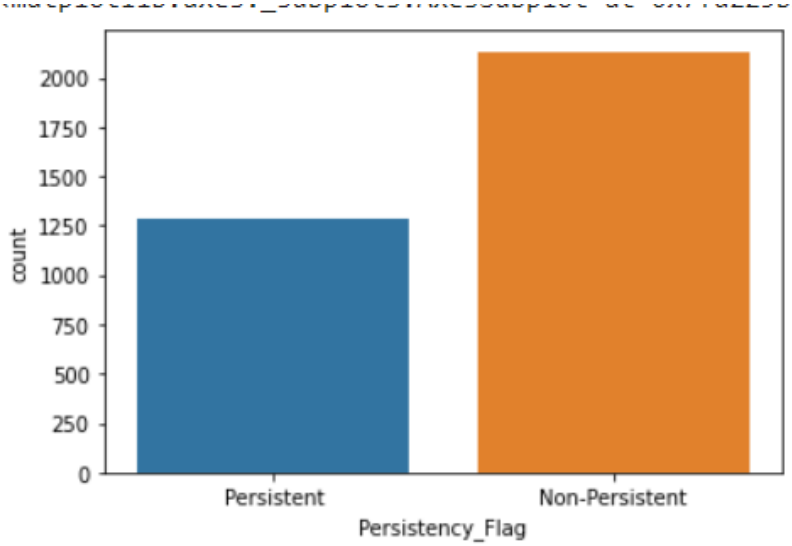
Table of Contents:

1. Problem Description

Persistence represents the time (e.g., days, months, years) over which a patient continues the treatment. For practical reasons, it might be assessed according to the time taken for a patient to fill their prescription and can capture both the timeliness and frequency of refilling. In reality, as defined by the adherence taxonomy, adherence is a dynamic behavior, consisting of initiation, implementation and discontinuation phases of treatment that vary over time, resulting in periods of persistence and non-persistence. Therefore, rather than measuring the specific components of adherence, we could measure persistence, which captures the chronology of adherence and enables us to examine and understand patterns of medication-taking behavior.
For that this project aims to build an automated classifier that predict whether a patient was persistent or not.

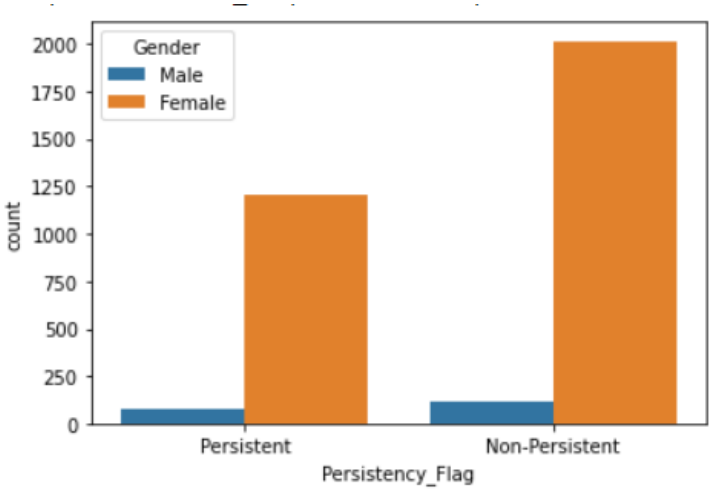
2. Exploratory data analysis

First let's start by visualizing the target variable: Persistency flag

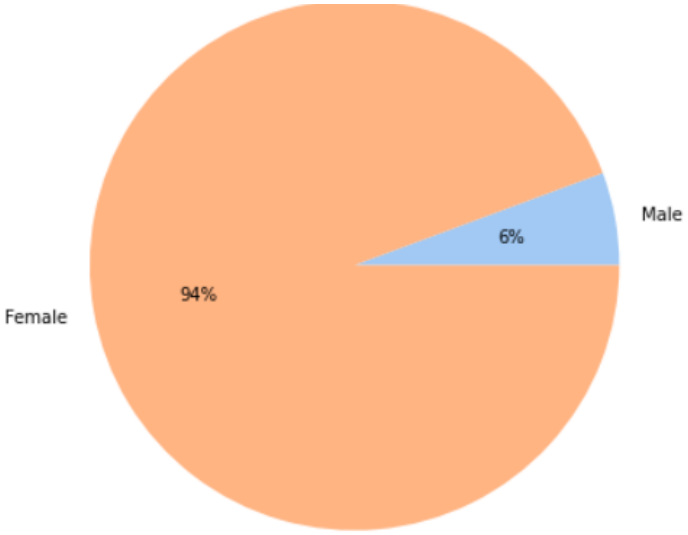


We notice that most of the patient are non-persistent in terms of data we can say that our classes are not balanced thing that should be handled in order to get good outcomes.

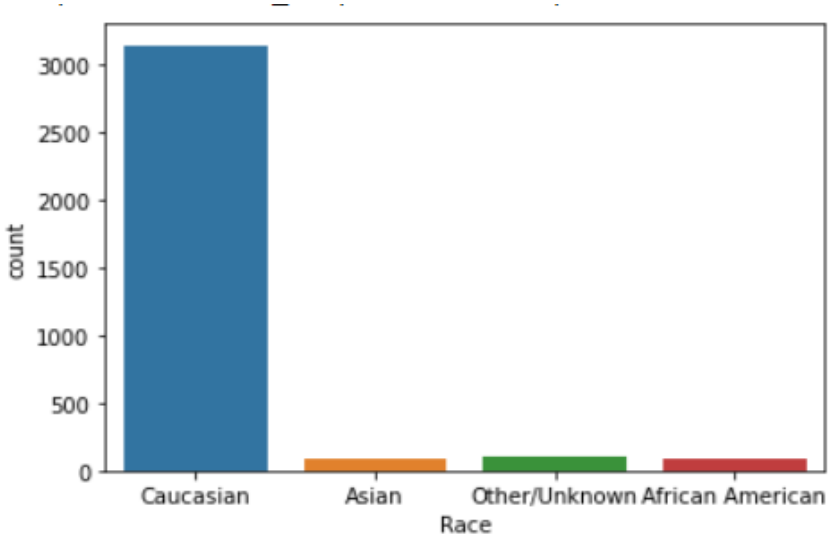
Now lets filter our target visualization by gender:



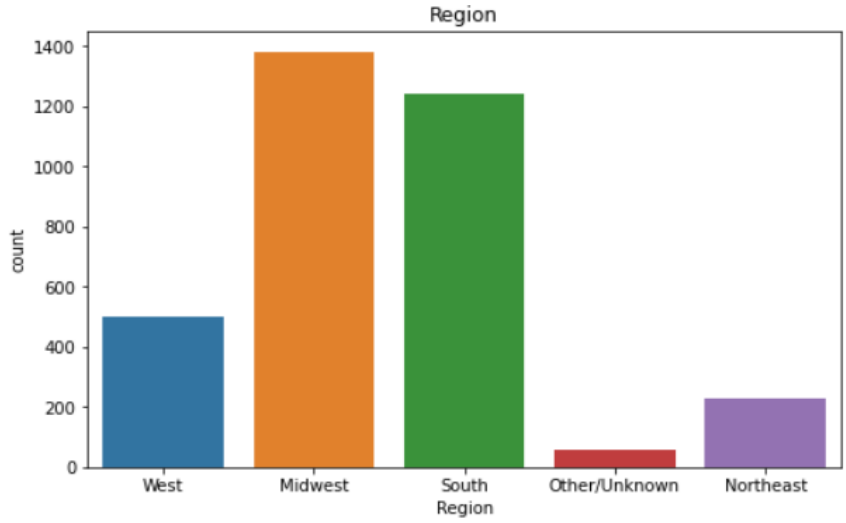
Here we see that the dominated gender in our data is female with a percentage of 94% from the totality of the data when males are presenting only 6% of data.



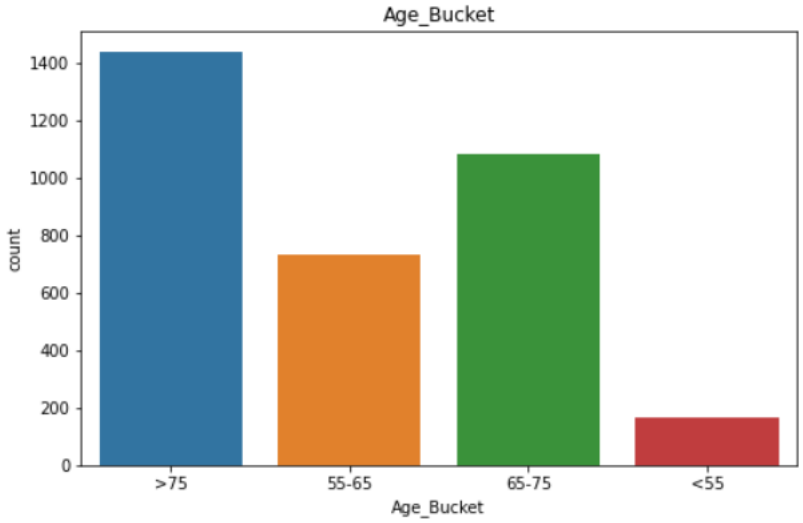
Now let's have an idea about the different patient's race:



The majority of patients have a Caucasian race. Then let's plot the different regions available in our data:



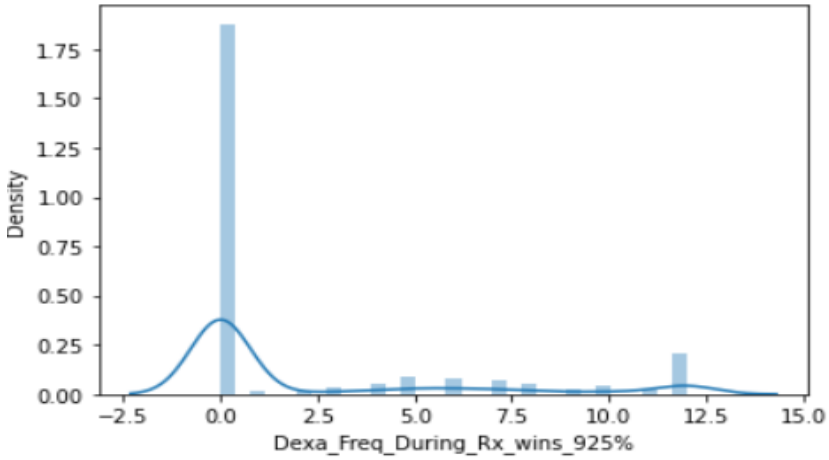
Here we can see the diversity of regions where Midwest and south region are the most Presented.



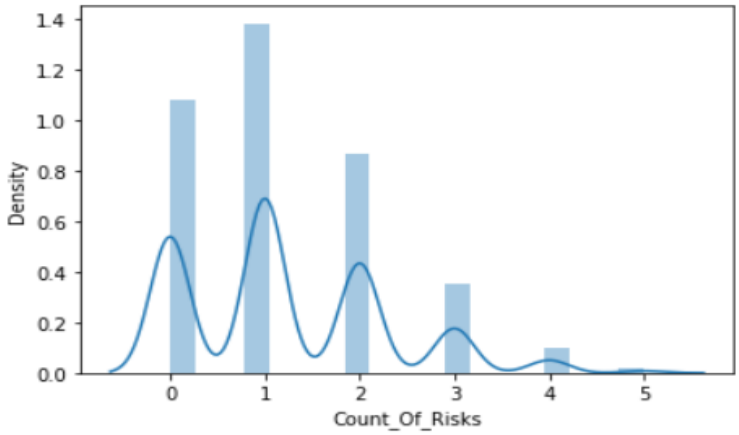
We constate from the plot above that the dataset is more focused on people who have more Than 55 which is normal related to the data topic.

The rest of categorical variables involve only two values which are N (no) and Y(yes) where N value occur more than Y .

Now let's move to continuous attributes where only two variables exist :



We notice that most of the values are presenting 0, this is due to winsorization technique that we've applied in order to handle the large number of outliers that we had at the beginning.



Here we see that we have 5 values where 1, 0 and 2 are the most presented. And the distribution doesn't seem to be normal and it's positively skewed.

Now let's move to correlation analysis and run some statistical tests in order to figure out the most important variables that have an impact on the target variable.
If we start by analyzing categorical variables versus categorical variable(target) we have to run a chi squared test since both of the variables are categorical

Null hypothesis: there is no relationship between the categorical variables.
Alternative hypothesis: there is a relationship between categorical variables.

```
y = df['Persistency_Flag']
important_var = []
for i in cat_list:

  contingency_table = pd.crosstab(df[i], y)
  print(contingency_table,"\n\n\n")
  chi2_stat,p_val,dof,ex = stats.chi2_contingency(contingency_table)
  print("CHI-SQUARE TEST VALUES")
  print("Chi Square Value : ",chi2_stat)
  print("Degree of Freedom : ",dof)
  print("P Value : ", p_val, "\n")
  if (p_val <= 0.05) :
    important_var.append(i)
    print(i, " has an impact on persistency flag \n\n")
  else:
    print(i, " doesn't affect persistency flag \n\n")
```

```
Persistency_Flag  Non-Persistent  Persistent
Gender
Female                      2015        1208
Male                         116          77


CHI-SQUARE TEST VALUES
Chi Square Value :  0.35575982921459065
Degree of Freedom :  1
P Value :  0.5508706006957192

Gender  doesn't affect persistency flag
```

After selecting all the categorical variables and gathering them in a list, I ran the chi squared test that will enable us to select only the variables correlated to the target variable.

```
#variables that have an impact on the target variable
important_var
```

```
['Region',
 'Ntm_Speciality',
 'Ntm_Specialist_Flag',
 'Ntm_Speciality_Bucket',
 'Gluco_Record_During_Rx',
 'Dexa_During_Rx',
 'Frag_Frac_During_Rx',
 'Risk_Segment_During_Rx',
 'Tscore_Bucket_During_Rx',
 'Change_T_Score',
 'Change_Risk_Segment',
 'Adherent_Flag',
 'Idn_Indicator',
 'Injectable_Experience_During_Rx',
 'Comorb_Encounter_For_Screening_For_Malignant_Neoplasms',
 'Comorb_Encounter_For_Immunization',
 'Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx',
 'Comorb_Vitamin_D_Deficiency',
 'Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified',
 'Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx'
 'Comorb_Long_Term_Current_Drug_Therapy',
 'Comorb_Dorsalgia',
 'Comorb_Personal_History_Of_Other_Diseases_And_Conditions',
 'Comorb_Other_Disorders_Of_Bone_Density_And_Structure',
 'Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias',
 'Comorb_Osteoporosis_without_current_pathological_fracture',
 'Comorb_Personal_history_of_malignant_neoplasm',
 'Comorb_Gastro_esophageal_reflux_disease',
 'Concom_Cholesterol_And_Triglyceride_Regulating_Preparations',
 'Concom_Narcotics',
```

```
#from 65 variables we get only 46 which are dependent of target variable
len(important_var)
```

```
46
```

For Continuous vs categorical variables since N > 30 and we have only 2 classes (persistent and non persistent) we will run a Z test, it assesses whether the average of two groups are statistically different from each other.

Null Hypothesis: There is no statistically difference between our variable(continuous) values for Various Class

Alternate Hypothesis: There is difference between Observed values for Various Class

```
num_var = df.select_dtypes(include = "int").columns
num_var = num_var.tolist()
num_var
```

```
['Dexa_Freq_During_Rx', 'Count_Of_Risks']
```

```
from statsmodels.stats import weightstats as stests

y = df['Persistency_Flag']
important_var_num = []

for i in num_var:
  ztest, pval = stests.ztest(y, df[i], alternative='two-sided')
  print("Z Test Value is ",ztest)
  print("P Value is ",pval)
  if (pval <= 0.05):
    important_var_num.append(i)
    print(i, " can be a good predictor to persistency flag \n\n")
  else:
    print(i, " doesn't affect persistency flag \n\n")
```

```
Z Test Value is  -26.189732657966882
P Value is  3.478806353132994e-151
Dexa_Freq_During_Rx  can be a good predictor to persistency flag


Z Test Value is  -42.41495396358258
P Value is  0.0
Count_Of_Risks  can be a good predictor to persistency flag
```

Based on the Z score results we get that both of the continuous variables have an impact on the persistency flag.

3. Recommendations

Based on this analysis I can recommend the elimination of some attributes that don't have any effect on the persistency flag which are:

['Region',
'Ntm_Speciality',
'Ntm_Specialist_Flag',
'Ntm_Speciality_Bucket',
'Gluco_Record_During_Rx',
'Dexa_During_Rx',
'Frag_Frac_During_Rx',
'Risk_Segment_During_Rx',
'Tscore_Bucket_During_Rx',
'Change_T_Score',
'Change_Risk_Segment',
'Adherent_Flag',
'Idn_Indicator',
'Injectable_Experience_During_Rx',
'Comorb_Encounter_For_Screening_For_Malignant_Neoplasms',
'Comorb_Encounter_For_Immunization',

'Comorb_Encntr_For_General_Exam_W_O_Complaint,_Susp_Or_Reprtd_Dx',
'Comorb_Vitamin_D_Deficiency',
'Comorb_Other_Joint_Disorder_Not_Elsewhere_Classified',
'Comorb_Encntr_For_Oth_Sp_Exam_W_O_Complaint_Suspected_Or_Reprtd_Dx',
'Comorb_Long_Term_Current_Drug_Therapy',
'Comorb_Dorsalgia',
'Comorb_Personal_History_Of_Other_Diseases_And_Conditions',
'Comorb_Other_Disorders_Of_Bone_Density_And_Structure',
'Comorb_Disorders_of_lipoprotein_metabolism_and_other_lipidemias',
'Comorb_Osteoporosis_without_current_pathological_fracture',
'Comorb_Personal_history_of_malignant_neoplasm',
'Comorb_Gastro_esophageal_reflux_disease',
'Concom_Cholesterol_And_Triglyceride_Regulating_Preparations',
'Concom_Narcotics',
'Concom_Systemic_Corticosteroids_Plain',
'Concom_Anti_Depressants_And_Mood_Stabilisers',
'Concom_Fluoroquinolones',
'Concom_Cephalosporins',
'Concom_Macrolides_And_Similar_Types',
'Concom_Broad_Spectrum_Penicillins',
'Concom_Anaesthetics_General',
'Concom_Viral_Vaccines',
'Risk_Rheumatoid_Arthritis',
'Risk_Untreated_Chronic_Hypogonadism',
'Risk_Smoking_Tobacco',
'Risk_Chronic_Malnutrition_Or_Malabsorption',
'Risk_Vitamin_D_Insufficiency',
'Risk_Poor_Health_Frailty',
'Risk_Excessive_Thinness',
'Risk_Immobilization',
'Dexa_Freq_During_Rx',
'Count_Of_Risks']

In total we keep only 49 independent variables instead of 67:

```
df_new = df[selected_var]
df_new.shape
```

```
(3416, 48)
```

4.Model Building:

After selecting the relevant feature we build our models based on them, this is the shape of the dataframe after feature selection :

```
df_new = df[selected_var]
df_new.shape
```

```
(3416, 48)
```

Then we transform the categorical variables into numerical using the one hot encoder transformer:

```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder(handle_unknown="ignore")
feature_arry = ohe.fit_transform(df_new[cat_list]).toarray()
col_names = ohe.get_feature_names_out()
col_names
```

In order to balance the target classes I used the smote function :

```
from imblearn.over_sampling import SMOTE
from collections import Counter

over_sampler = SMOTE(k_neighbors=2)
x_smote, y_smote = over_sampler.fit_resample(x, y)
print(f"Training target statistics: {Counter(y_smote)}")
```

```
Training target statistics: Counter({1: 2131, 0: 2131})
```

4.1. Linear model:

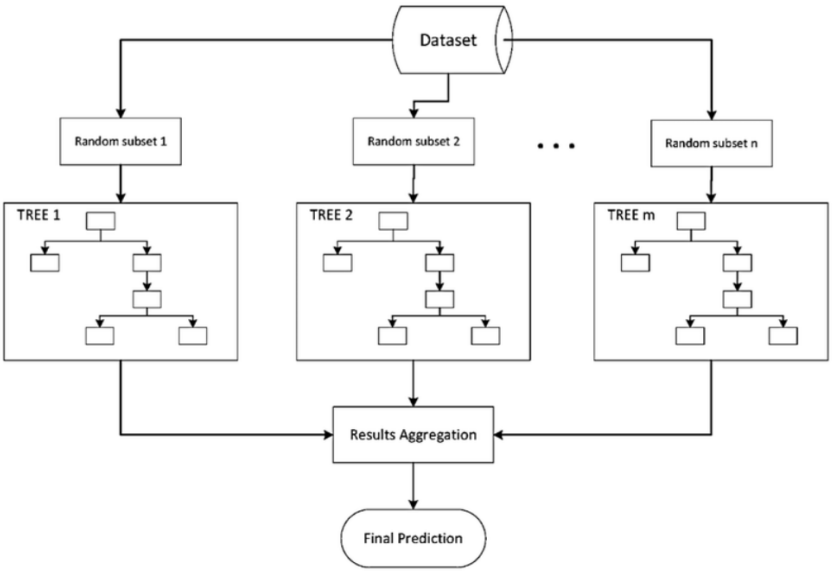
For linear model I used SVC, or Support Vector Classifier which is a supervised machine learning algorithm typically used for classification tasks. SVC works by mapping data points to a high-dimensional space and then finding the optimal hyperplane that divides the data into two classes.with `SVC(C=1, gamma=0.1)` are the Best parameters selected using the gridsearchCV method from model selection in sckit learn :

```
model = SVC()
param_grid = {'C': [0.1,1, 10, 100], 'gamma': [1,0.1,0.01,0.001],'kernel': ['rbf', 'poly', 'sigmoid']}
grid = GridSearchCV(model,param_grid,refit=True,verbose=2)
grid.fit(x_train, y_train)
grid.score(x_test, y_test)
grid.best_estimator_
```

In order to get better model performance I used k fold cross validation with k=10, and also to overcome the overfitting issue since we are dealing with small dataset.

4.2. Bagging:

Ensemble methods is a machine learning technique that combines several base models in order to produce one optimal predictive model. One of it's type is Bagging which gets its name because it combines Bootstrapping and Aggregation to form one ensemble model. Given a sample of data, multiple bootstrapped subsamples are pulled. A Decision Tree is formed on each of the bootstrapped subsamples. After each subsample Decision Tree has been formed, an algorithm is used to aggregate over the Decision Trees to form the most efficient predictor. Here is an example to explain the technique:



In our case the base estimator is SVC used in the linear model part .
Another bagging model is executed : Random forest classifier with the best parameters obtained from the
gridsearchCV method are :

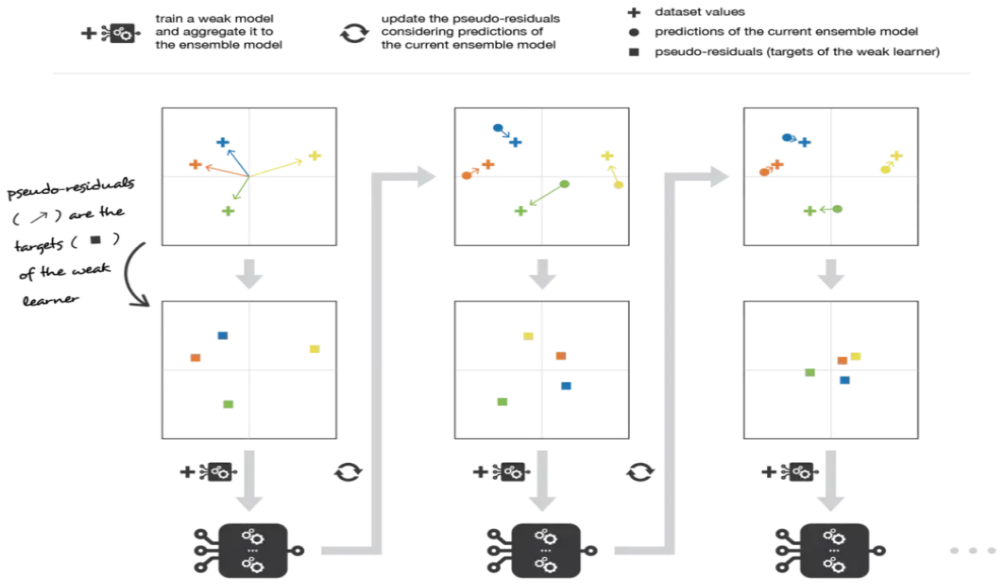
```
[ ]  from sklearn.ensemble import RandomForestClassifier

     rfc=RandomForestClassifier(random_state=42)
     param_grid = {
         'n_estimators': [200, 500],
         'max_features': ['auto', 'sqrt', 'log2'],
         'max_depth' : [4,5,6,7,8],
         'criterion' :['gini', 'entropy']
     }
     CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
     CV_rfc.fit(x_train, y_train)
     print(CV_rfc.best_params_)
```

```
{'criterion': 'gini', 'max_depth': 8, 'max_features': 'auto', 'n_estimators': 200}
```

4.3. Boosting:

boosting, that often considers homogeneous weak learners, learns them sequentially in a very adaptative way (a base model depends on the previous ones) and combines them following a deterministic strategy.

Gradient boosting:
In gradient boosting, the ensemble model we try to build is also a weighted sum of weak learners



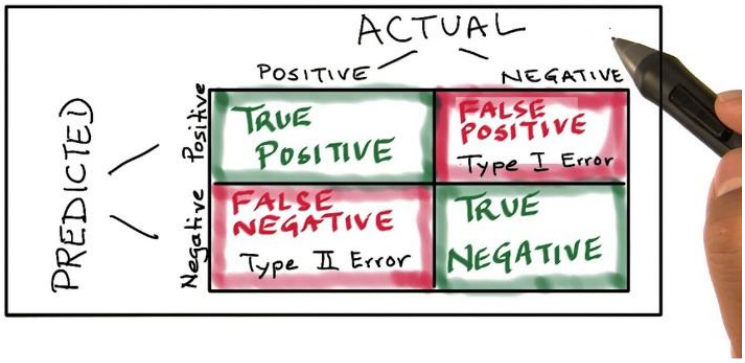
Now let's get into the evaluations of each model

5.Model evaluation:

Model evaluation is the process of using different evaluation metrics to understand a machine learning model's performance, as well as its strengths and weaknesses. Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

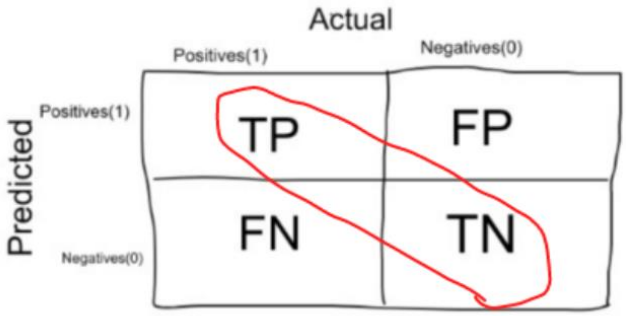
1. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known



2. Accuracy

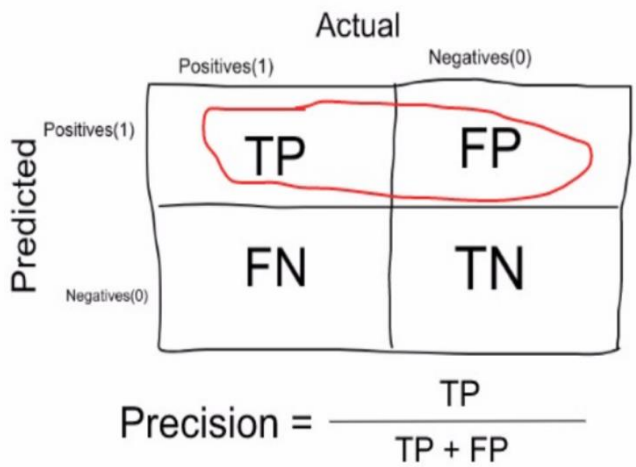
Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made



$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

3. Precision

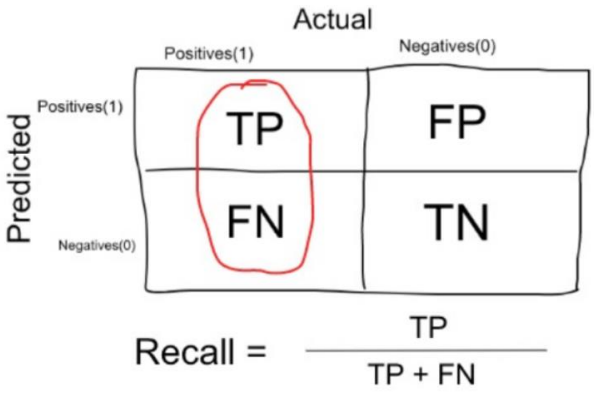
Precision is defined as the number of true positives divided by the number of true positives plus the number of false positives. Precision is about being precise



$$\text{Precision} = \frac{TP}{TP + FP}$$

4. Recall

When it is actually the positive result, how often does it predict correctly



5. Auc - Roc curve

AUC-ROC curve is a performance measurement for classification problem at various thresholds settings. ROC is a probability curve and AUC represents degree or measure of separability.
It tells how much model is capable of distinguishing between classes.
Higher the AUC, better the model is at predicting 0s and 1s as 1as. By analogy, Higher the AUC, better the model is at distinguishing between survived and not

For that I grouped all the models result in a dataframe to facilitate the comparison process :

	model	Accuracy	Precision	Recall	Roc-Auc
0	SVC	85.802139	86.550060	84.610849	93.113084
1	SVC bagging	85.890374	86.530367	84.846698	92.939398
2	Random forest classifier	83.221666	85.768501	79.952830	91.460714
3	GradientBoosting classifier	83.748749	85.776942	80.719340	92.190443

We conclude that SVC and SVC bagging are reaching the highest performance in accuracy, precision, recall and Roc-Auc.