Data Glacier

Your Deep Learning Partner

Project: Data Science Healthcare - Persistency of a drug

Data science Virtual Internship

FATIMA EZZAHRA AMRAOUI

30-November-2021

Agenda

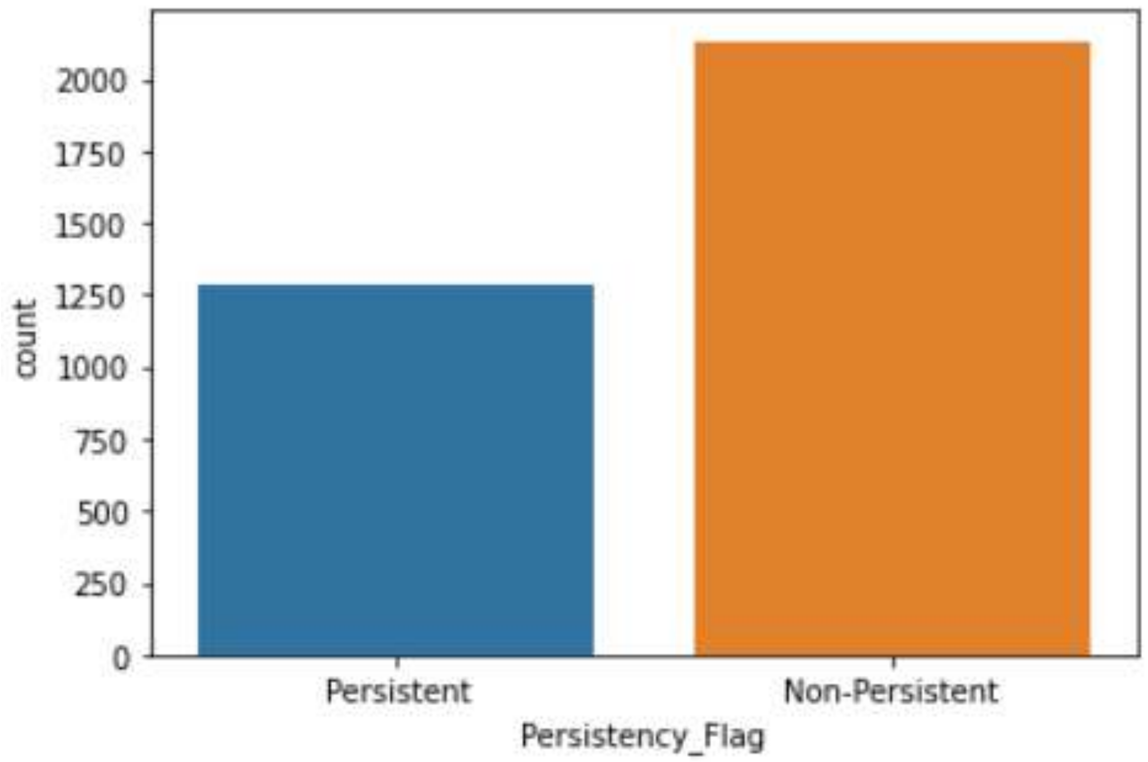
Problem Description

- One of the challenge for all Pharmaceutical companies of is to understand the persistency of drug as per the physician prescription.

- The purpose of this work is to build a classifier for the given dataset to predict whether a patient is persistent or not.

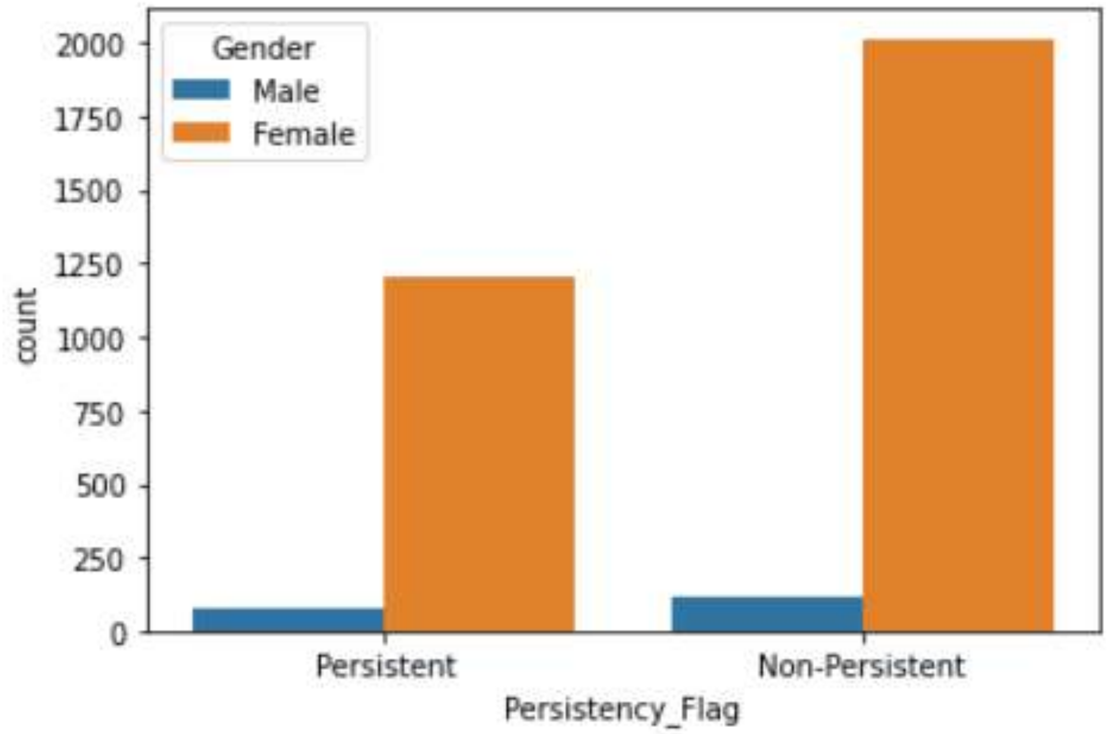
EDA

- **Categorical var :** 67 (including target var)
- **Continious var :** 2



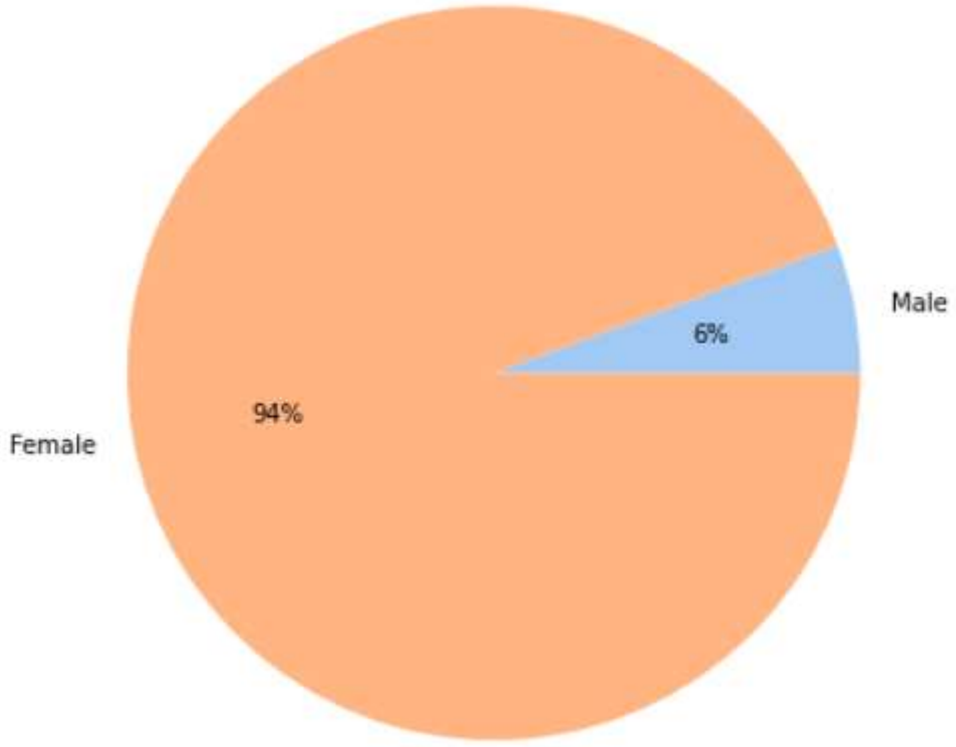
- Non-persistent class is higher than persistent one (unbalanced classes)

EDA



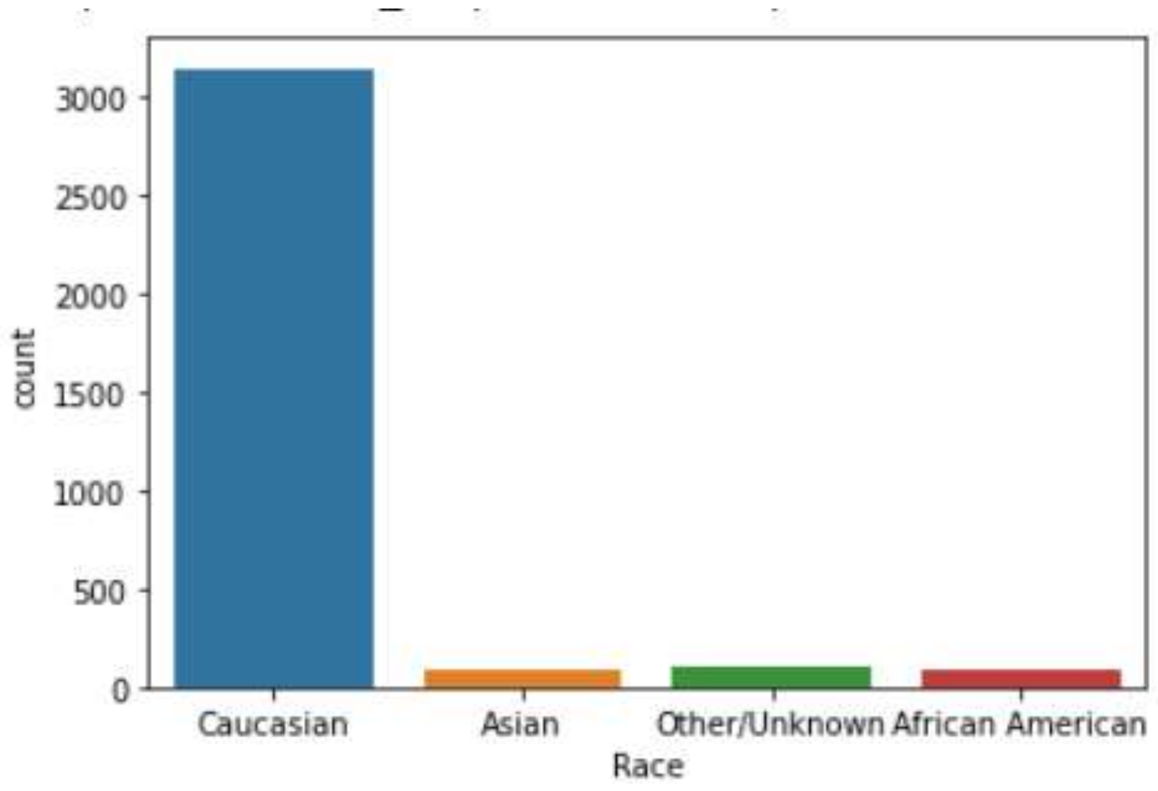
- the dominated gender in our data is female

EDA



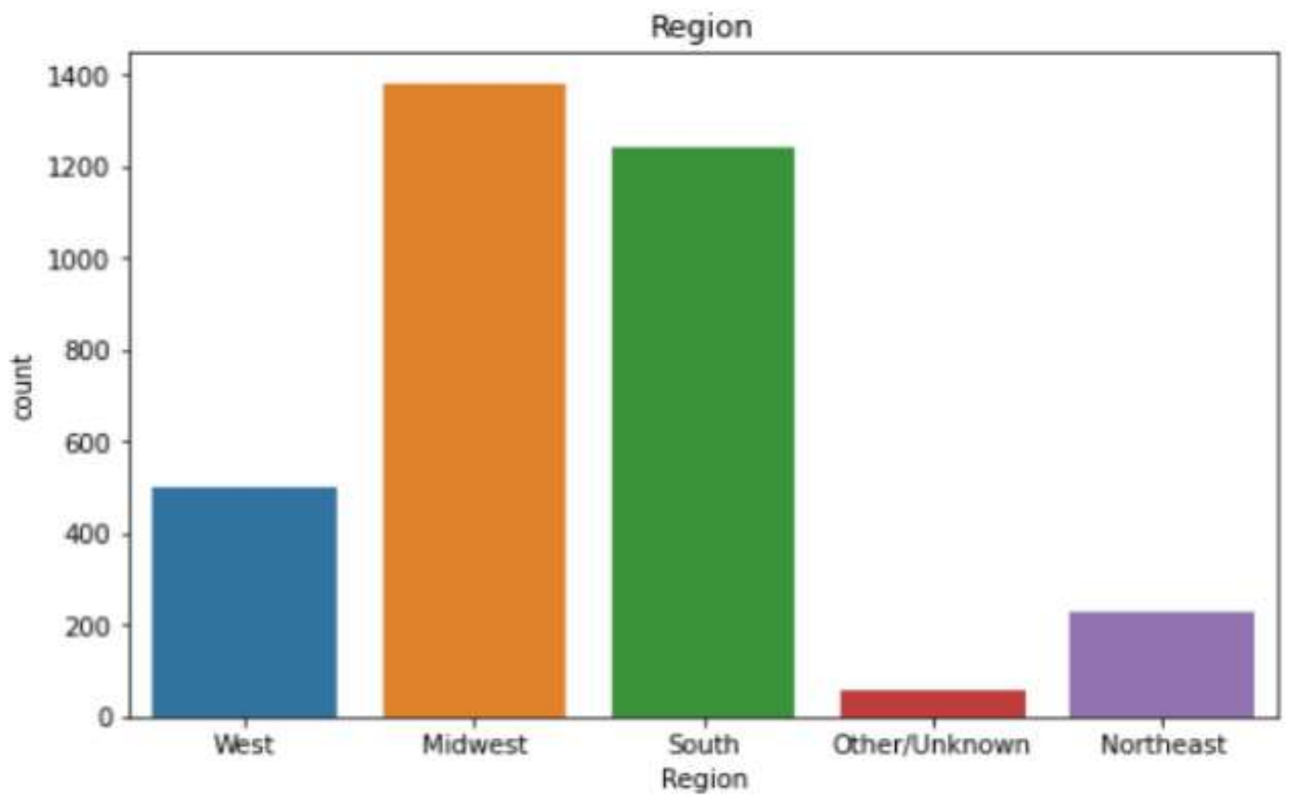
- Here we see that the dominated gender in our data is female with a percentage of 94% from the totality of the data when males are presenting only 6% of data.

EDA



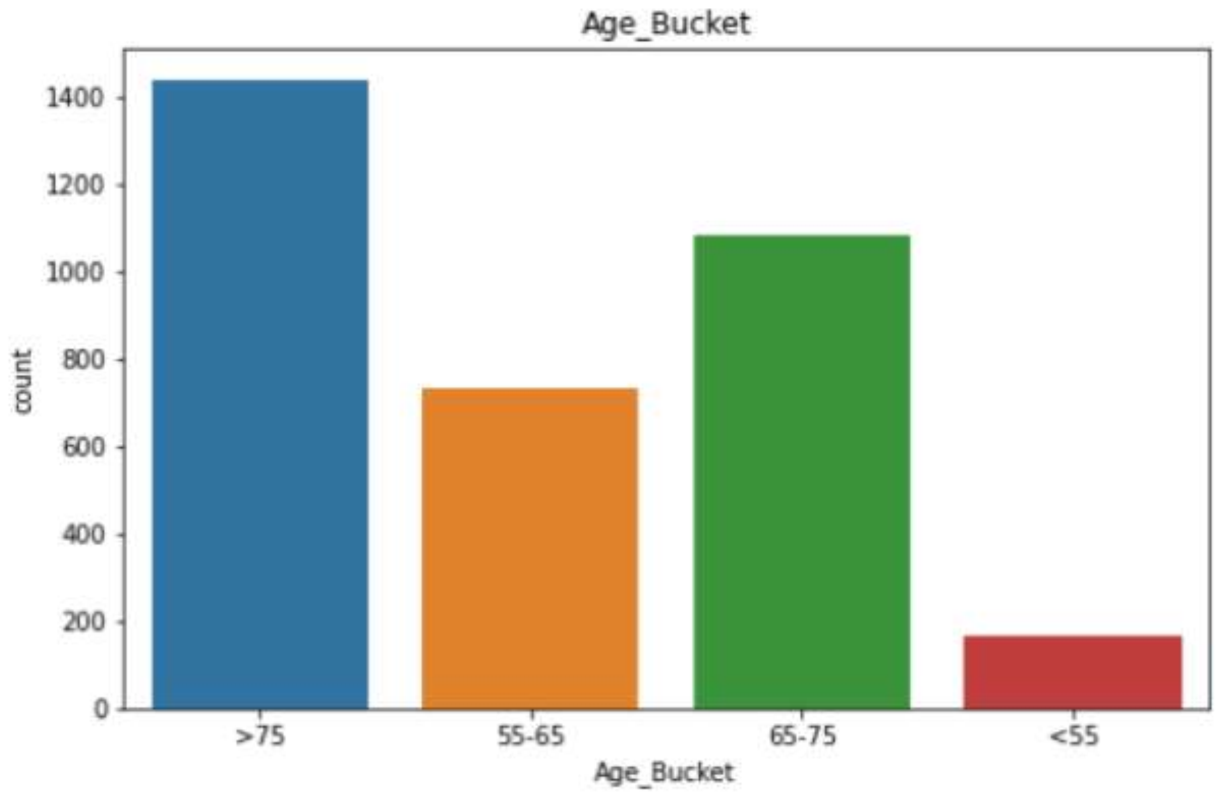
- The majority of patients have a Caucasian race.

EDA



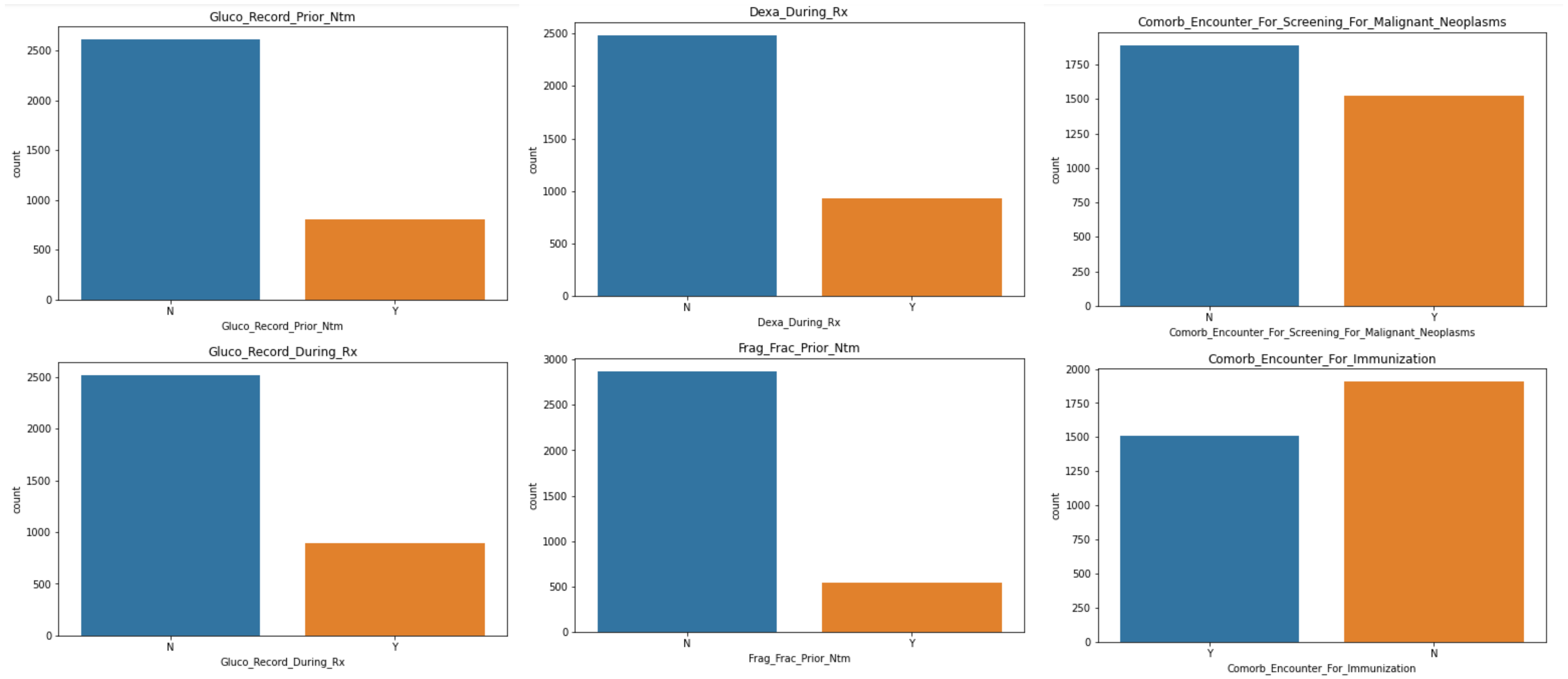
- Here we can see the diversity of regions where Midwest and south region are the most Presented.

EDA



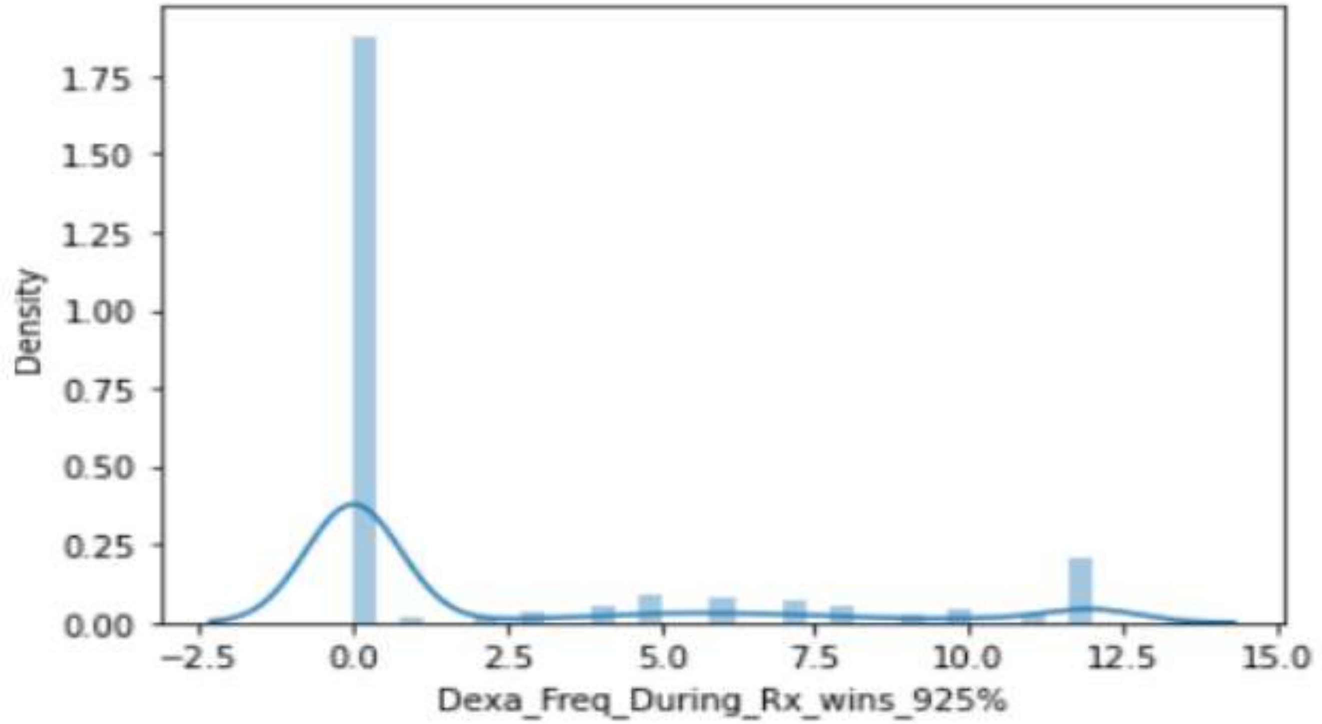
- the dataset is more focused on people who have more Than 55

EDA



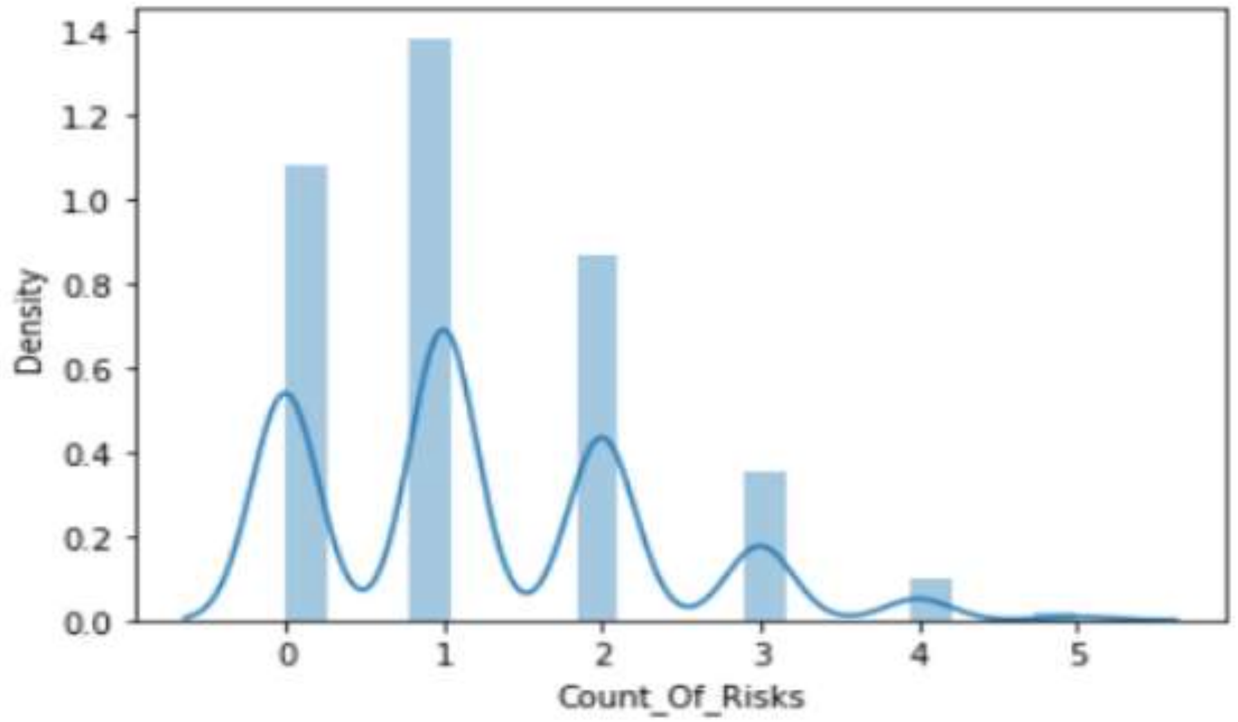
- Most of the rest of categorical variables involve two values which are N (no) and Y(yes) where N value occur more than Y (with some exceptions of course)

EDA



- 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.

EDA



- 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.

EDA

Bivariate analysis : Categorical vs categorical

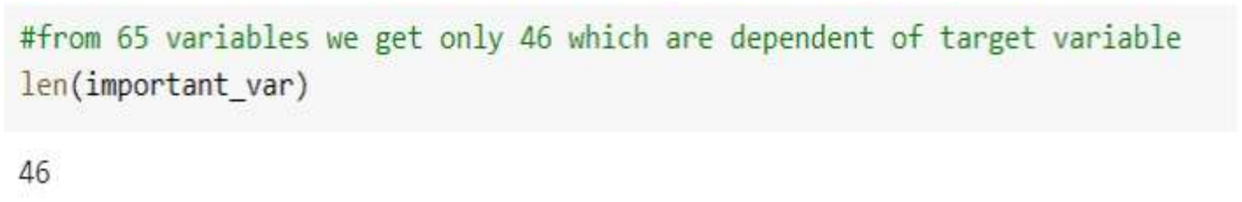
Chi

squared test :

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

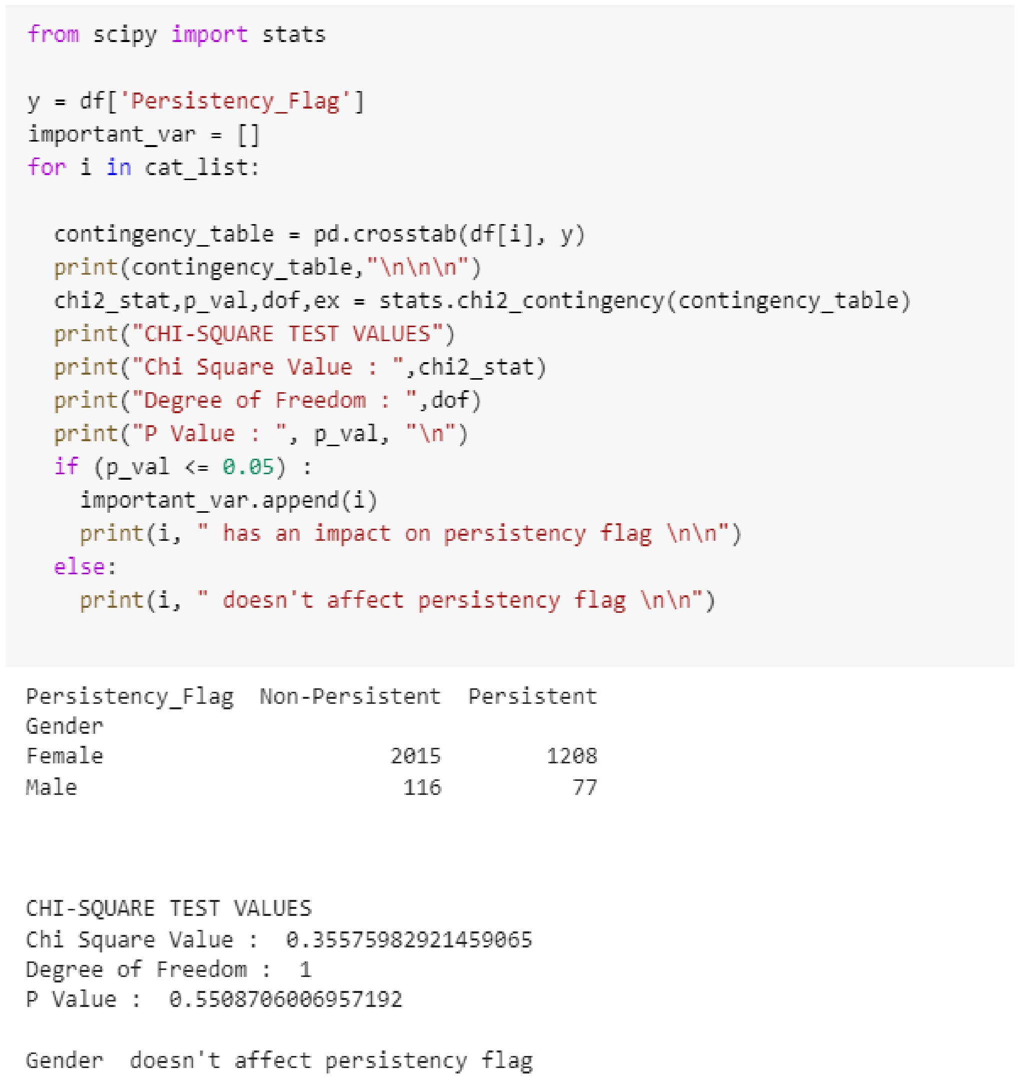
Alternative hypothesis: there is a relationship between categorical variables.

- *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*



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

```
46
```



```
from scipy import stats

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
```

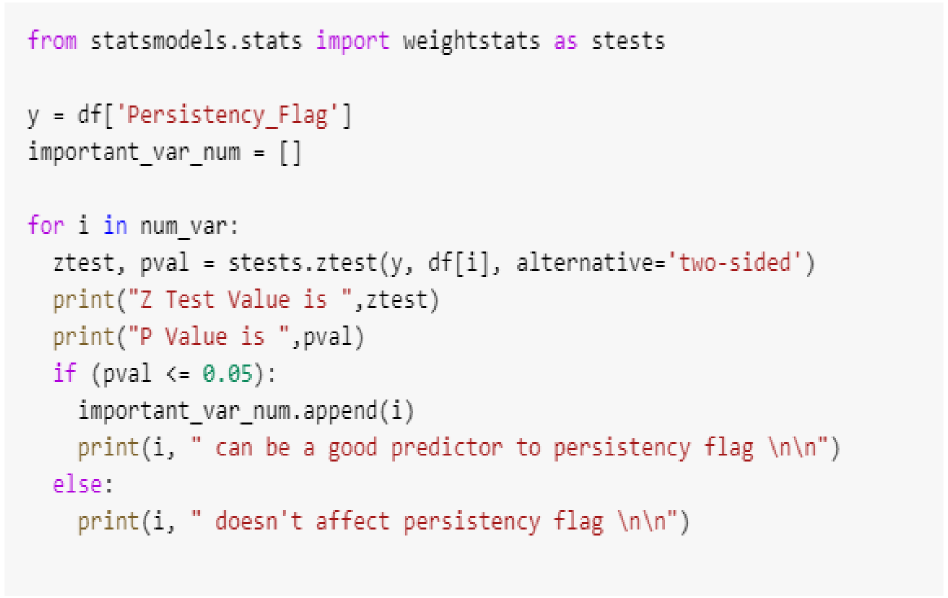
EDA

Bivariate analysis : Categorical vs Continious

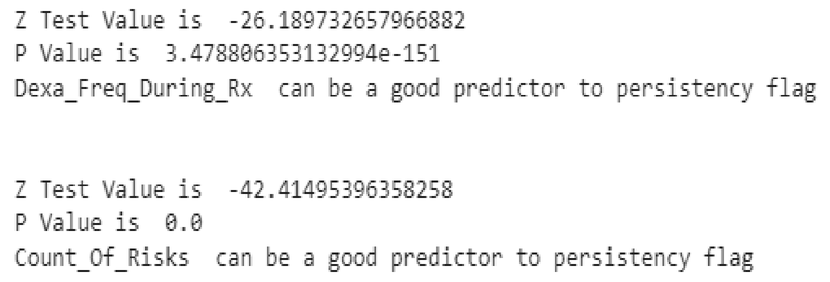
<u>Z score test :</u>

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



```
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")
```

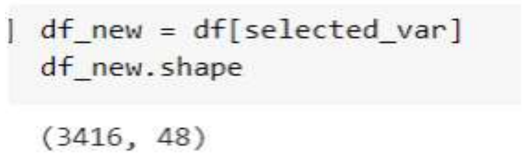


```
Z Test Value is  -26.189732657966882
P Value is  3.478806353132994e-151
Dexa_Freq_During_Rx  can be a good predictor to persistency flag


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.*

EDA

In total we keep only 49 independent variables instead of 67



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

```
(3416, 48)
```

```
['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']
```

Model Recommendation

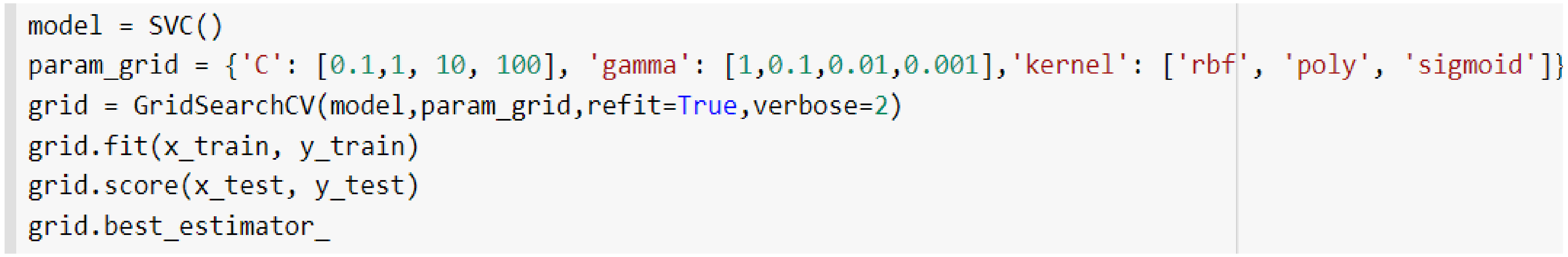
- Apply a technique to balance the Target variable classes (such as smooth function).
- Use algorithmes which can control overfitting(L1 & L2 regularization…) since our data is small such as :

  -Logistic Regression
  -XGBOOST…

- We can also try other classifier : SVC,RFC…

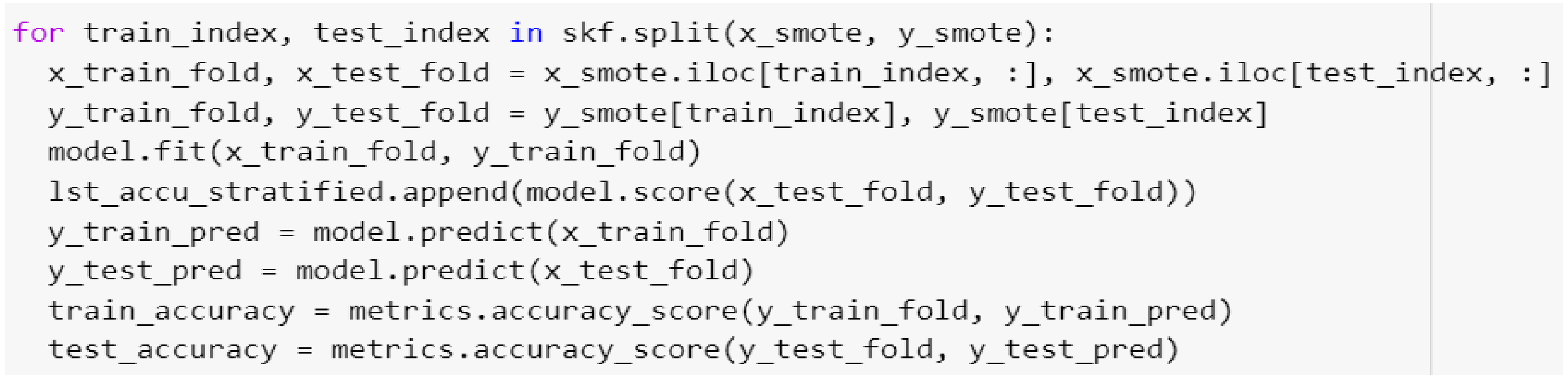
Model building

- Linear model : SVC



```
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_
```

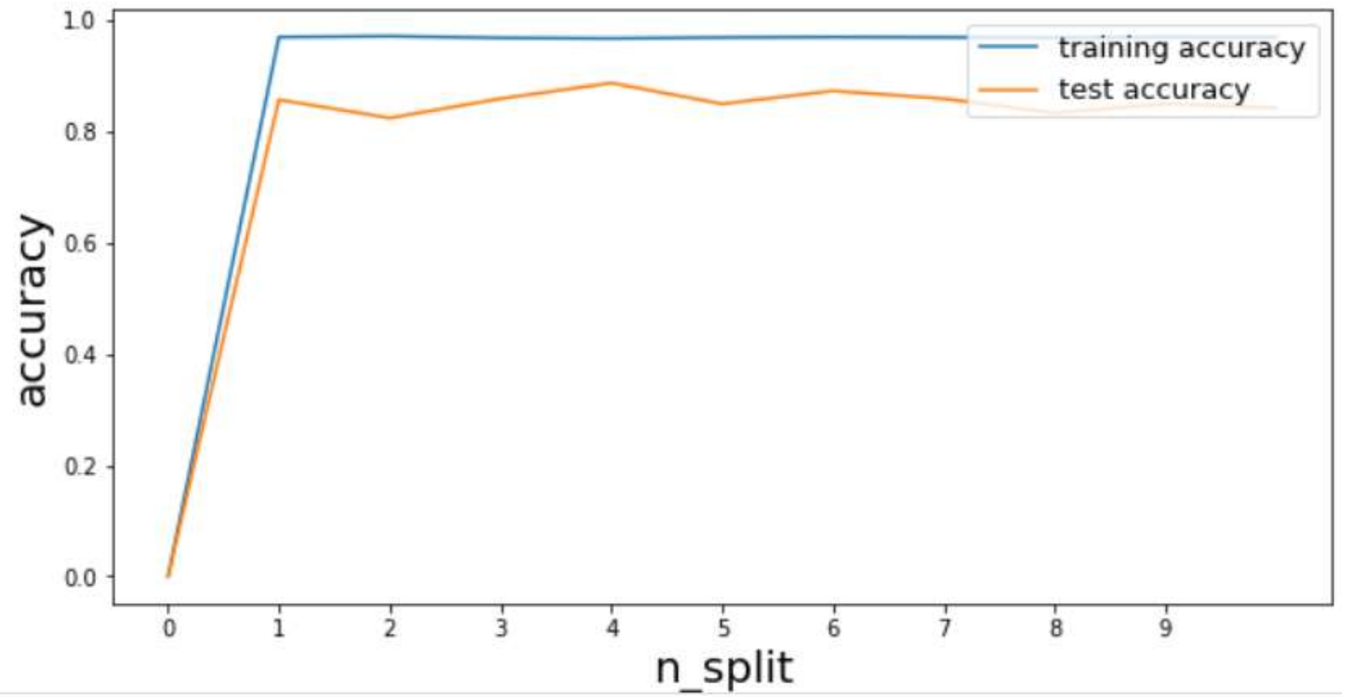
- Best parameters : `SVC(C=1, gamma=0.1)`



```
for train_index, test_index in skf.split(x_smote, y_smote):
  x_train_fold, x_test_fold = x_smote.iloc[train_index, :], x_smote.iloc[test_index, :]
  y_train_fold, y_test_fold = y_smote[train_index], y_smote[test_index]
  model.fit(x_train_fold, y_train_fold)
  lst_accu_stratified.append(model.score(x_test_fold, y_test_fold))
  y_train_pred = model.predict(x_train_fold)
  y_test_pred = model.predict(x_test_fold)
  train_accuracy = metrics.accuracy_score(y_train_fold, y_train_pred)
  test_accuracy = metrics.accuracy_score(y_test_fold, y_test_pred)
```

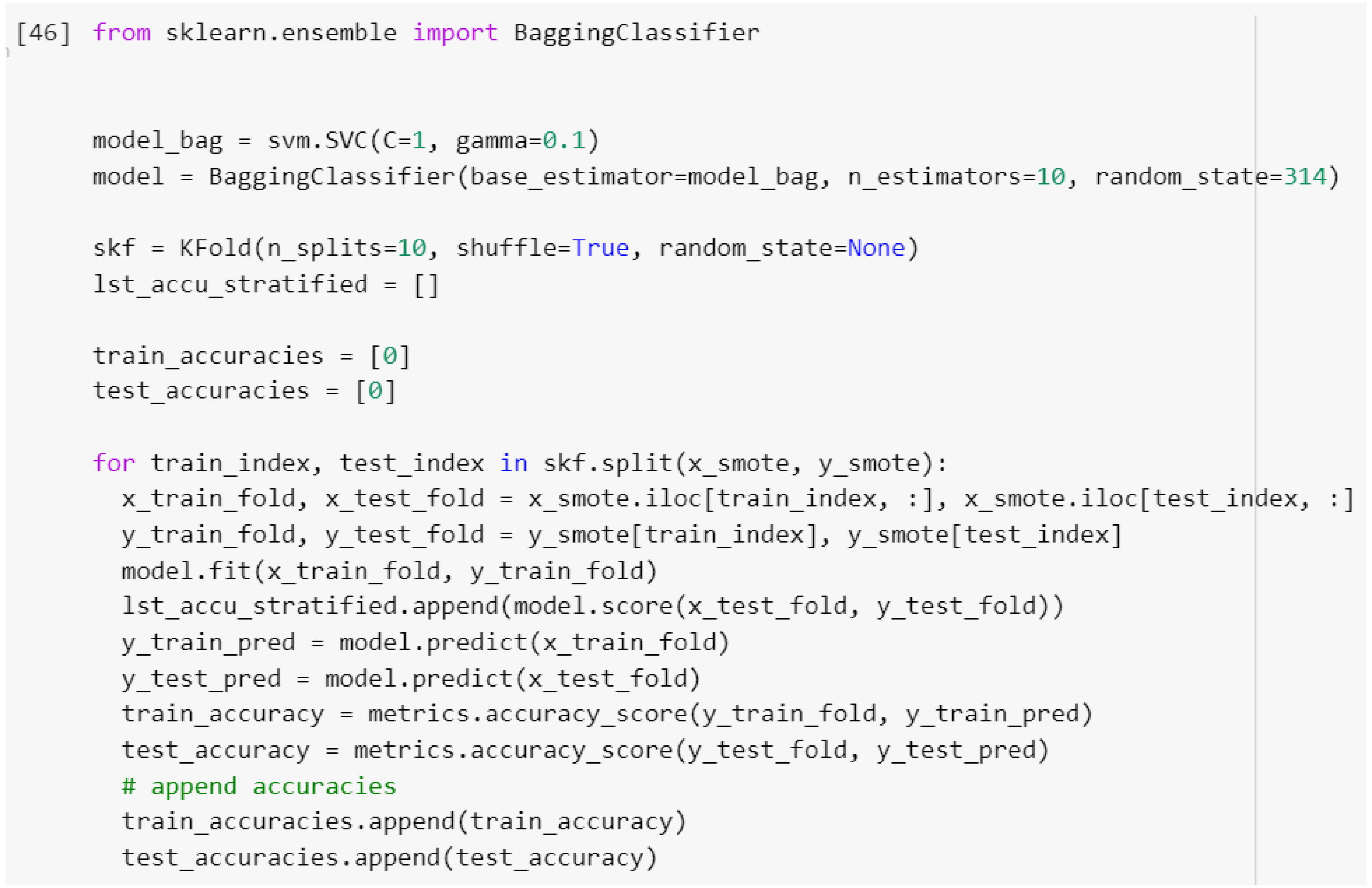
Model building

- The overfitting is acceptable



Model building

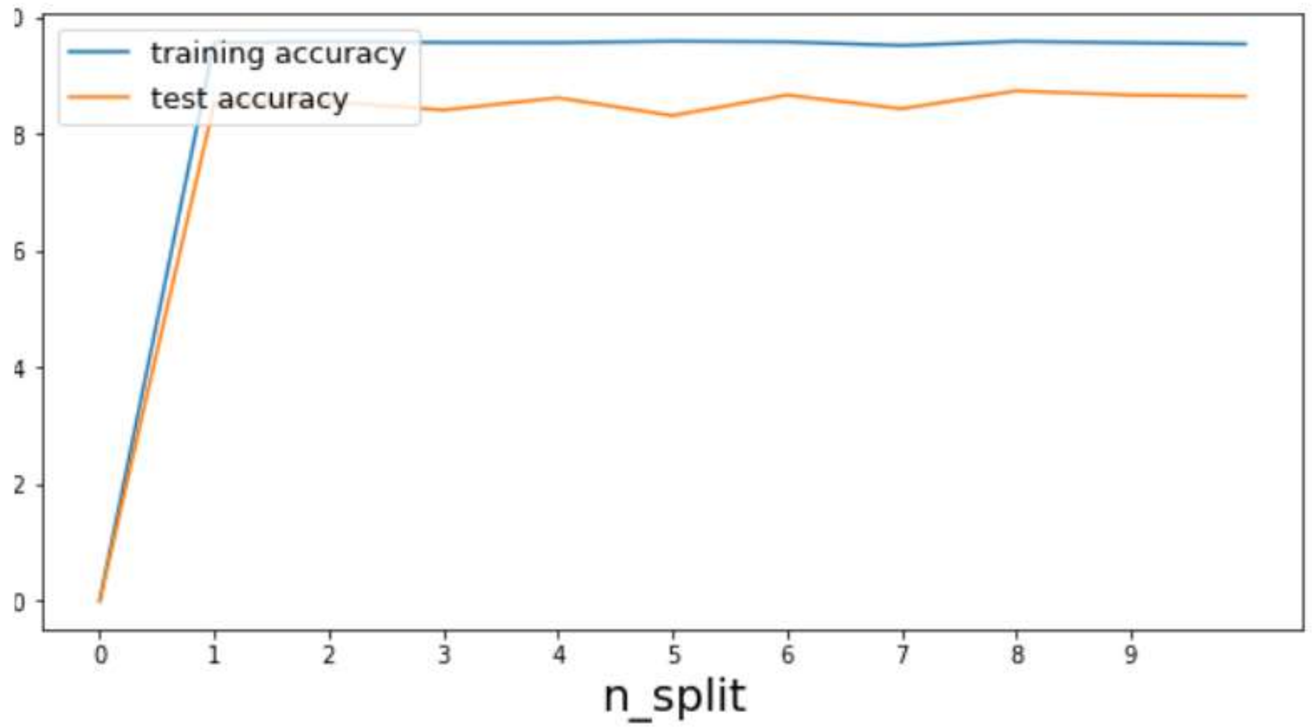
- Ensemble method:Bagging
- Using K-folf cross validation



```
[46] from sklearn.ensemble import BaggingClassifier


     model_bag = svm.SVC(C=1, gamma=0.1)
     model = BaggingClassifier(base_estimator=model_bag, n_estimators=10, random_state=314)

     skf = KFold(n_splits=10, shuffle=True, random_state=None)
     lst_accu_stratified = []

     train_accuracies = [0]
     test_accuracies = [0]

     for train_index, test_index in skf.split(x_smote, y_smote):
       x_train_fold, x_test_fold = x_smote.iloc[train_index, :], x_smote.iloc[test_index, :]
       y_train_fold, y_test_fold = y_smote[train_index], y_smote[test_index]
       model.fit(x_train_fold, y_train_fold)
       lst_accu_stratified.append(model.score(x_test_fold, y_test_fold))
       y_train_pred = model.predict(x_train_fold)
       y_test_pred = model.predict(x_test_fold)
       train_accuracy = metrics.accuracy_score(y_train_fold, y_train_pred)
       test_accuracy = metrics.accuracy_score(y_test_fold, y_test_pred)
       # append accuracies
       train_accuracies.append(train_accuracy)
       test_accuracies.append(test_accuracy)
```

Model building

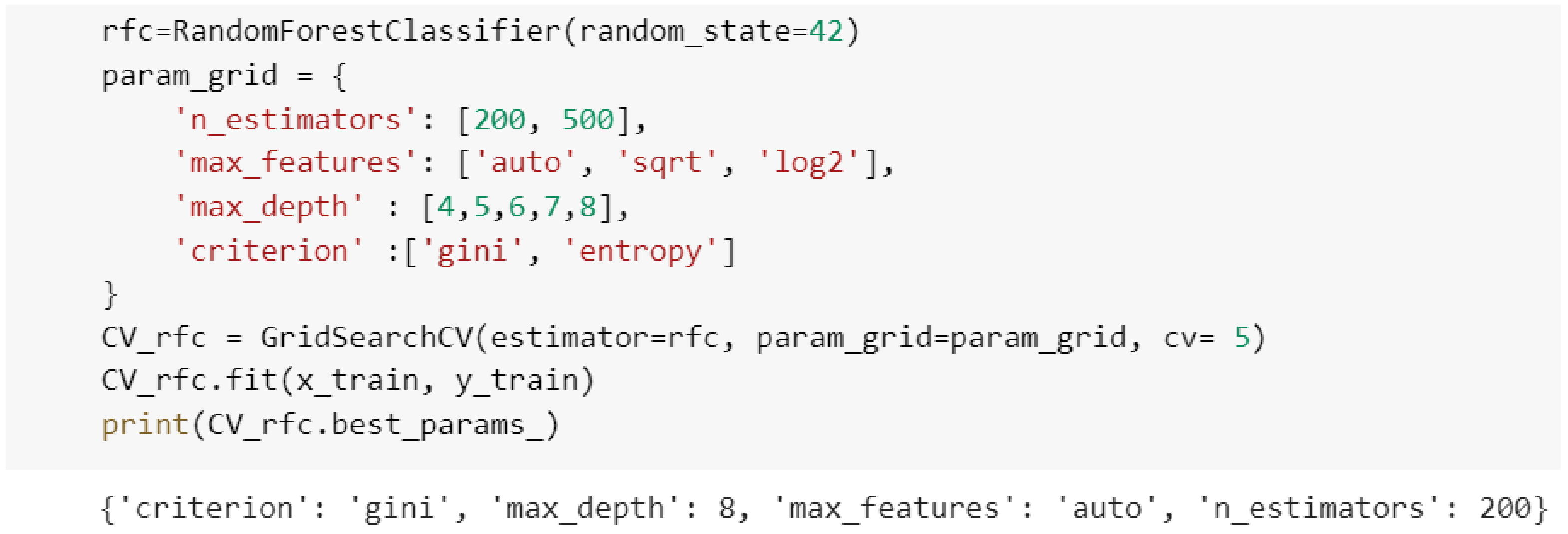
- The overfitting is acceptable



Model building

- Random forest classifier(bagging)

GridsearchCV:

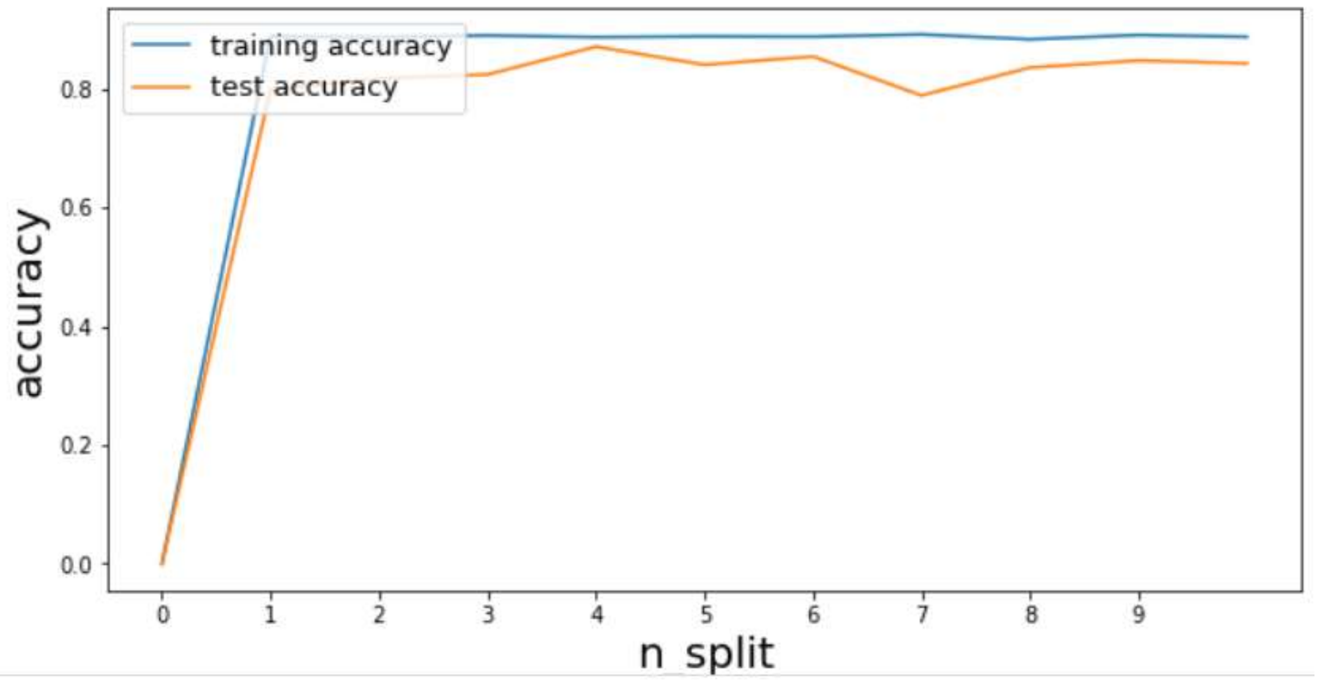


```
rfc=RandomForestClassifier(random_state=42)
param_grid = {
    'n_estimators': [200, 500],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth' : [4,5,6,7,8],
    'criterion' :['gini', 'entropy']
}
CV_rfc = GridSearchCV(estimator=rfc, param_grid=param_grid, cv= 5)
CV_rfc.fit(x_train, y_train)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

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

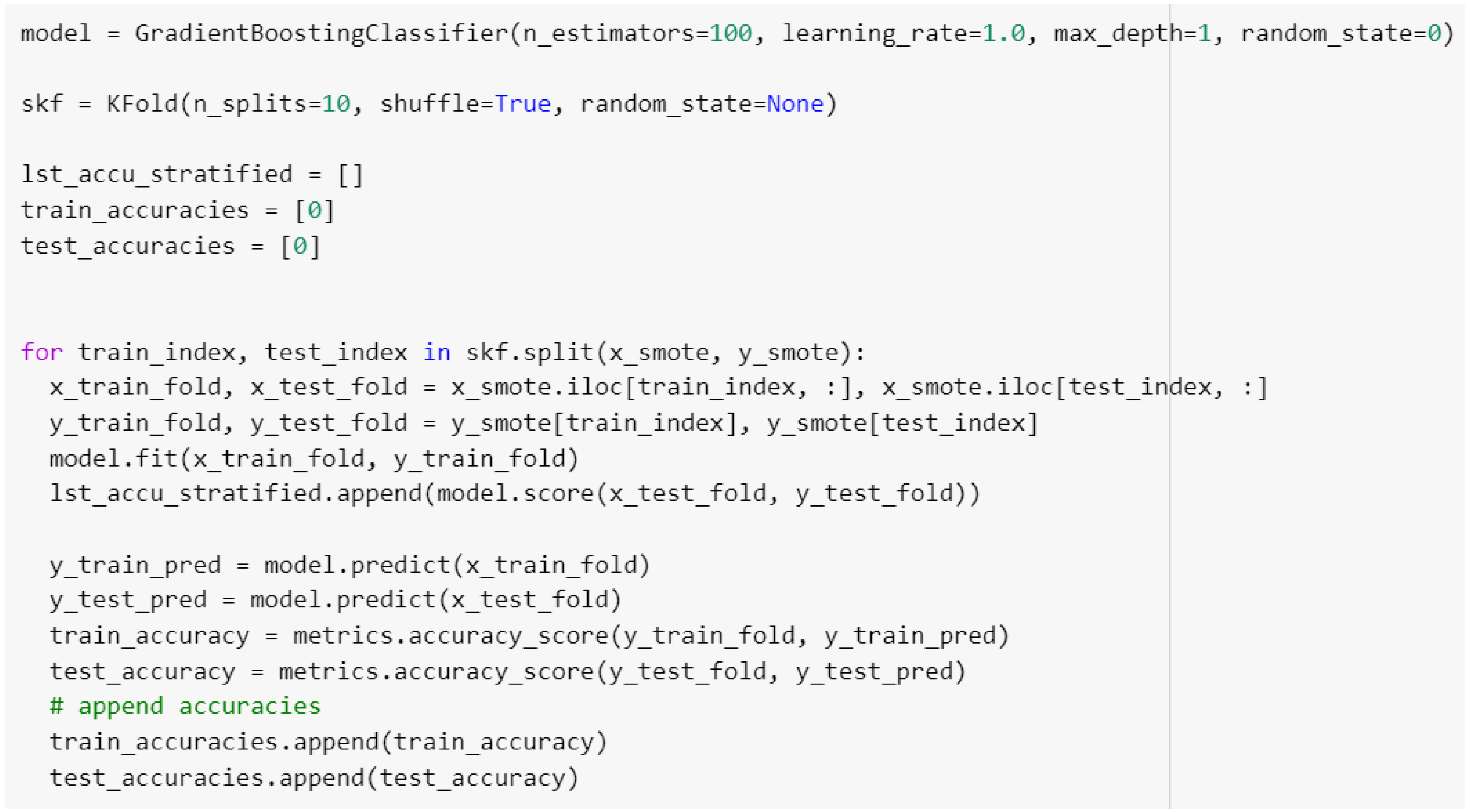
Model building

- The overfitting is less in random forest than SVC:



Model building

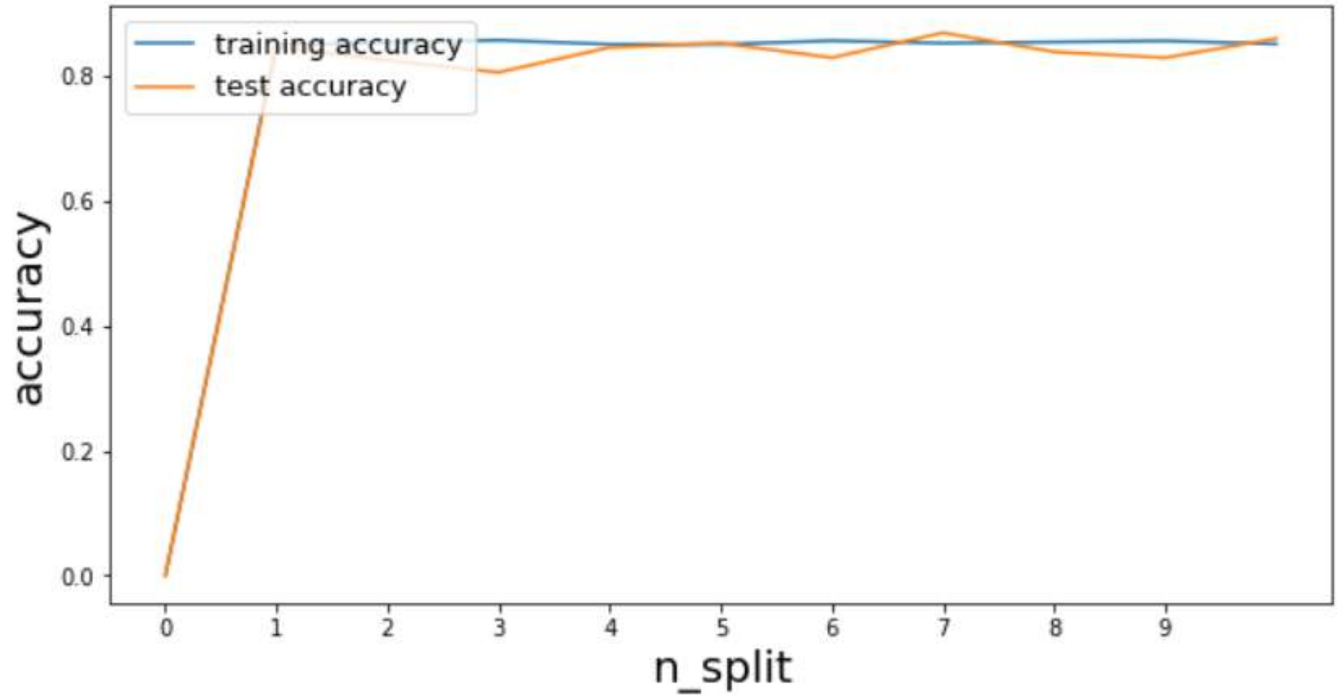
- Boostig : Gradient Boosting classifier:



```
model = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0, max_depth=1, random_state=0)

skf = KFold(n_splits=10, shuffle=True, random_state=None)

lst_accu_stratified = []
train_accuracies = [0]
test_accuracies = [0]


for train_index, test_index in skf.split(x_smote, y_smote):
  x_train_fold, x_test_fold = x_smote.iloc[train_index, :], x_smote.iloc[test_index, :]
  y_train_fold, y_test_fold = y_smote[train_index], y_smote[test_index]
  model.fit(x_train_fold, y_train_fold)
  lst_accu_stratified.append(model.score(x_test_fold, y_test_fold))

  y_train_pred = model.predict(x_train_fold)
  y_test_pred = model.predict(x_test_fold)
  train_accuracy = metrics.accuracy_score(y_train_fold, y_train_pred)
  test_accuracy = metrics.accuracy_score(y_test_fold, y_test_pred)
  # append accuracies
  train_accuracies.append(train_accuracy)
  test_accuracies.append(test_accuracy)
```

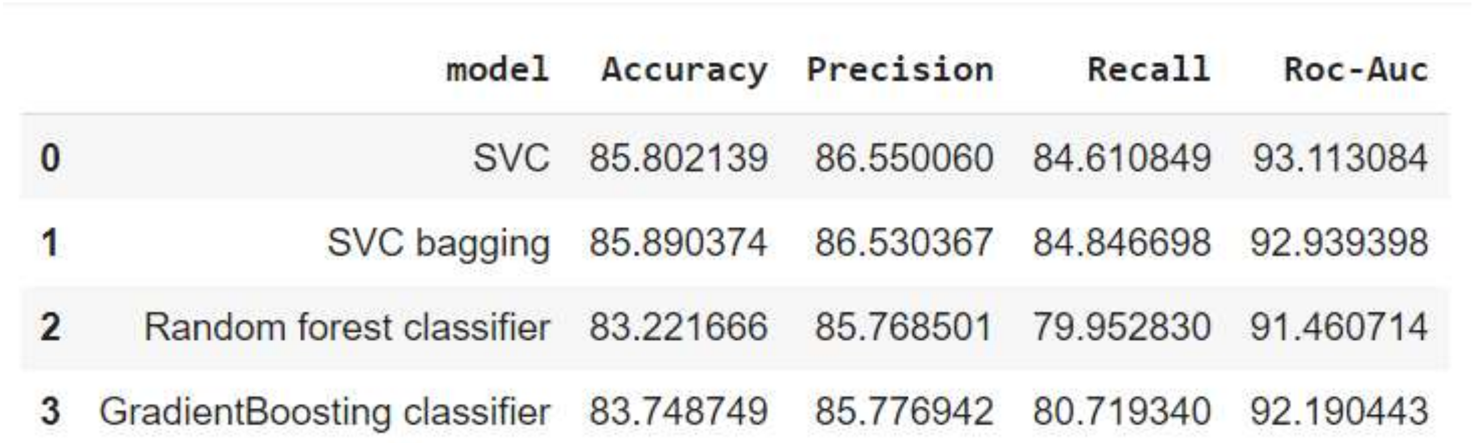
Model building

- The curve shows that gradient boosting is better than all previous model in terms of overfitting:



Model evaluation

SVC perform better than other models in accuracy, precision, recall and Roc-Auc with 85.55%, 86.55%, 84.61%
And 93.11% respectively



	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

Thank You

Data Glacier

Your Deep Learning Partner