Smart Lead Scoring Engine

Goal : Predicting 0 or 1 indicating if the user will buy the product in next 3 months or not

ARVIND BAWANKAR

Agenda

Introduction

- A D2C startup develops products using cutting edge technologies like Web 3.0. Over the past few months, the company has started multiple marketing campaigns offline and digital both. As a result, the users have started showing interest in the product on the website. These users with intent to buy product(s) are generally known as leads (Potential Customers).

- Objective is to build predictive model to classify if the user would buy the product in the next 3 months or not.

1. Setup Environment

The goal of this section is to:

Import all the packages

Set the options for data visualizations

Importing necessary libraries, we used to in our entire process

```
# Data Manipulation
import numpy as np
import pandas as pd
import math

# Data Visualization
import seaborn as sns
import matplotlib.pyplot as plt

# to divide train and test set
from sklearn.model_selection import train_test_split

#Handling Class Imbalance
from   imblearn.over_sampling import RandomOverSampler
```

```
# to build the models
from xgboost import XGBClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier,
ExtraTreesClassifier, GradientBoostingClassifier

# to evaluate the models
from   sklearn.model_selection import train_test_split,
GridSearchCV
from   sklearn.metrics import confusion_matrix ,
classification_report, accuracy_score, roc_auc_score,
plot_roc_curve, precision_recall_curve,
plot_precision_recall_curve
```

2. Data Overview

The initial step is to understand the data set we are dealing with, for this we have overview
Our data, so that we can learn more about there behavior which help us
To build the predictive model with high accuracy

Purpose is to:

1. Load the datasets
2. Explore the features

Load the train and test data using pd.read_csv()

Load Data using pd.read_csv()

After loading data with pd.read_csv() we observe the below insight about our data

Observation :

- Train data set contains 39161 rows and 19 variables i.e feature
- Test data set contain 13184 rows and 18 columns i.e feature
- means train data has 18 feature and one target, here "buy" has the target variable

Data Description :

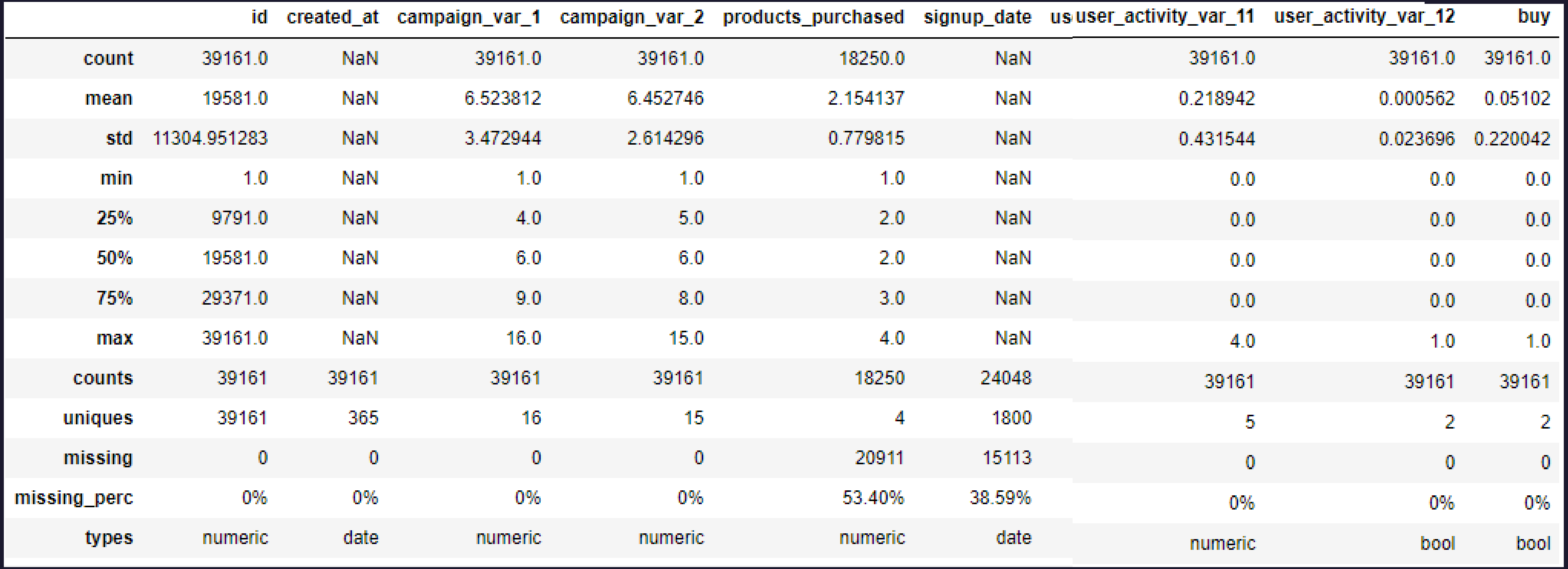
Column Name	Description
id	**Unique identifier of a lead**
created_at	**Date of lead dropped**
signup_date	**Sign up date of the user on the website**
campaign_var (1 and 2)	**Campaign information of the lead**
products_purchased	**No. of past products purchased at the time of dropping the lead**
user_activity_var (1to 12)	Derived activities of the user on the website
buy	0 or 1 indicating if the user will buy the product in next 3 months or not (Target Variable)

3. Basic Data Stats¶

The goal of this section is to:

- ➢ - Get the dimensions of data
- ➢ - Get the summary of data
- ➢ - Get various statistics of data

Summary of dataframe (5 number summary of dataframe)



	id	created_at	campaign_var_1	campaign_var_2	products_purchased	signup_date	us user_activity_var_11	user_activity_var_12	buy
count	39161.0	NaN	39161.0	39161.0	18250.0	NaN	39161.0	39161.0	39161.0
mean	19581.0	NaN	6.523812	6.452746	2.154137	NaN	0.218942	0.000562	0.05102
std	11304.951283	NaN	3.472944	2.614296	0.779815	NaN	0.431544	0.023696	0.220042
min	1.0	NaN	1.0	1.0	1.0	NaN	0.0	0.0	0.0
25%	9791.0	NaN	4.0	5.0	2.0	NaN	0.0	0.0	0.0
50%	19581.0	NaN	6.0	6.0	2.0	NaN	0.0	0.0	0.0
75%	29371.0	NaN	9.0	8.0	3.0	NaN	0.0	0.0	0.0
max	39161.0	NaN	16.0	15.0	4.0	NaN	4.0	1.0	1.0
counts	39161	39161	39161	39161	18250	24048	39161	39161	39161
uniques	39161	365	16	15	4	1800	5	2	2
missing	0	0	0	0	20911	15113	0	0	0
missing_perc	0%	0%	0%	0%	53.40%	38.59%	0%	0%	0%
types	numeric	date	numeric	numeric	numeric	date	numeric	bool	bool

Observation:

- column products_purchased and signup_date found missing values of 53.4% and 38.59% respectively
- campaign_var (1 and 2) has numeric values and found 15 to 16 unique values in range of(1 to 16)
- products_purchased has also numeric values and have 4 unique values
- user_activity_var (1 to 12) - has some numeric values and some boolean values
- Target class is imbalanced.will do the necessory action if need

4. Data Preprocessing for EDA¶

preprocessing to feed the correct data to the model to learn and predict.

Model performance depends on the quality of data feeded to the model to train

The goal of this section is to:

- ➢ Identifying missing values in data
- ➢ Replacing missing values in data
- ➢ Removing missing values from data
- ➢ Handling Skewed data
- ➢ Outliers detection and removal
- ➢ Finding and fixing Imbalance class
- ➢ Encoding Categorical Data

After doing the Data Preprocessing below observation found

1. Let's check the variable types, and found no categorical variable
2. Number of Numerical Variable : 16
3. Missing values :
    - ✓ products_purchased (53.39%)
    - ✓ signup_date (38.59%)

Checking Relationship between missing data on products_purchased and buy found some interesting insight

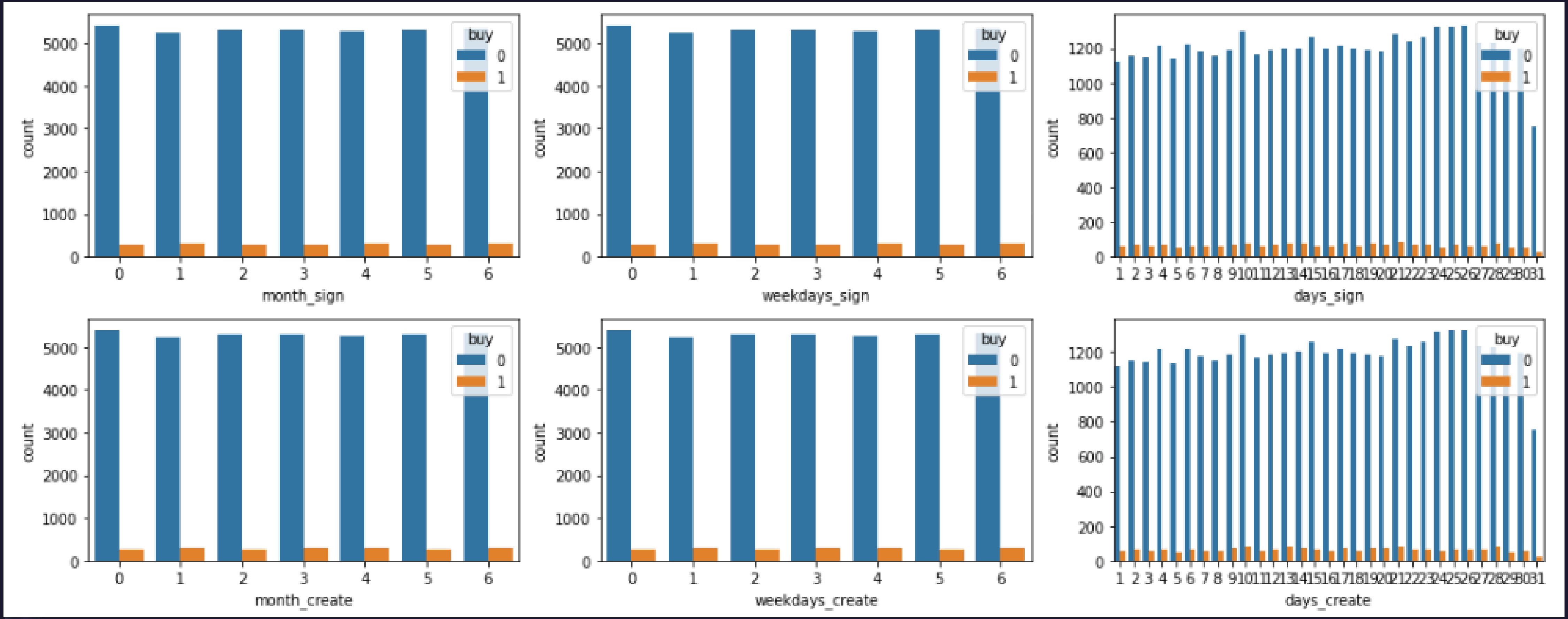
5% has the potential customer who bought the product
95% customer do not bought the product

Replacing 5% null values with 1 and 95% null values with 0

In submission file all null values replaced with 0 as we have no target yet

Numerical variables: 16

We have two datetime variable, let's create some date related feature like month, weekdays, and days and see the impact on Target and see the visual



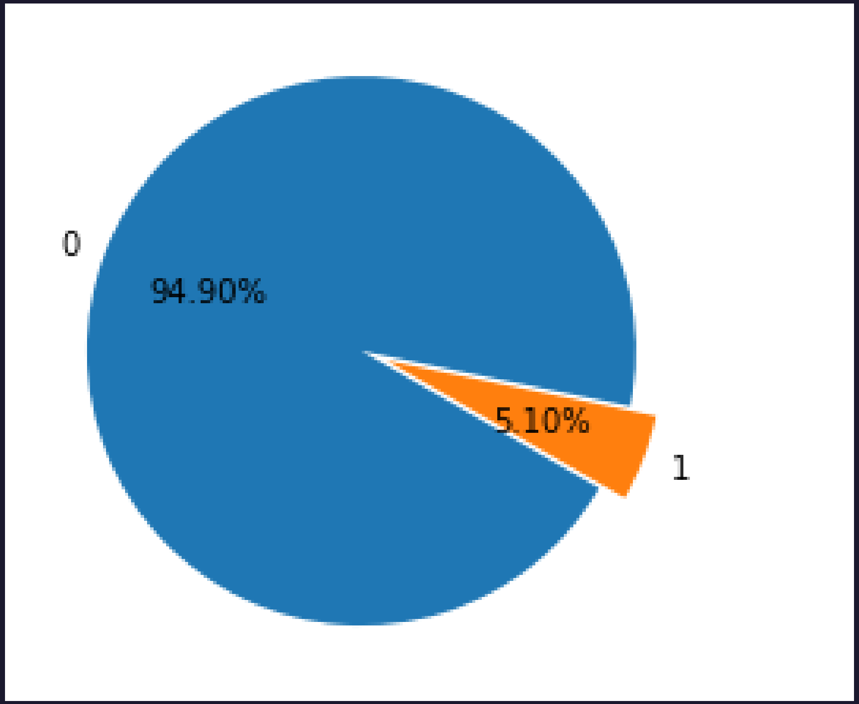
5. Exploratory Data Analysis¶

Exploratory data analysis investigate data sets to find out patterns and see if any of the variables can be useful to explain / predict the target variables.

The goal of this section is to:

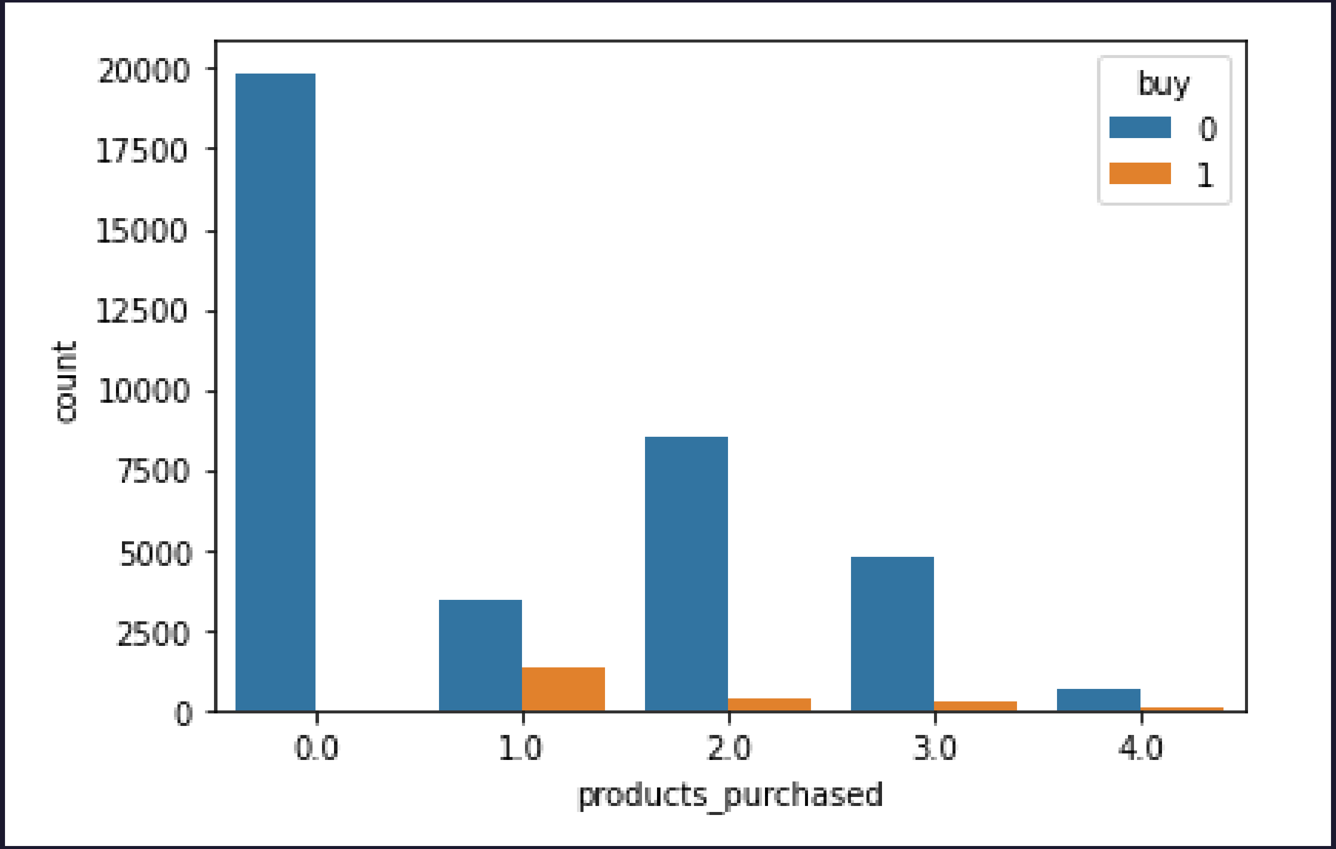
- Check if the target variable is balanced or is there a need to balance the target variableReplacing missing values in data
- Get insights or relationships from the data which would be useful from business perspectiveHandling Skewed data

Check distribution of target variable



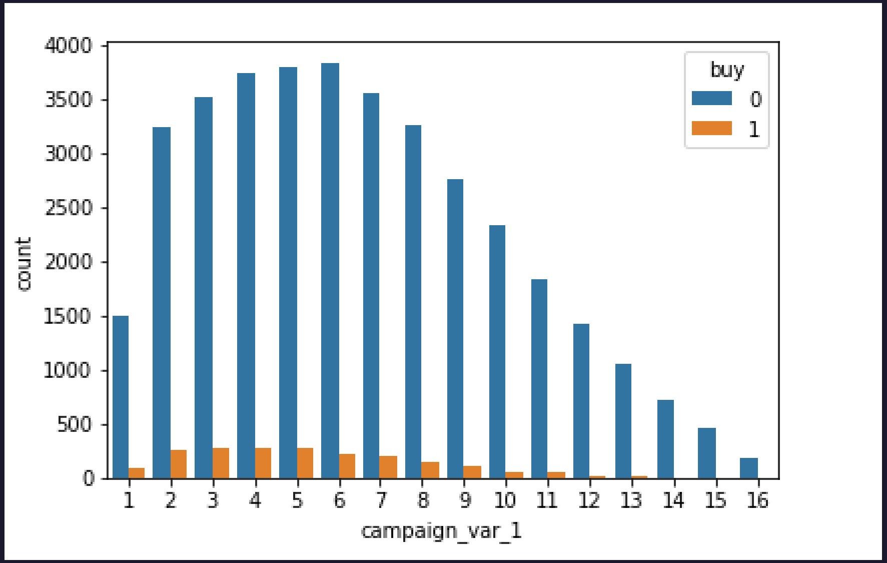
Target class is imbalanced. will do the necessary action if need

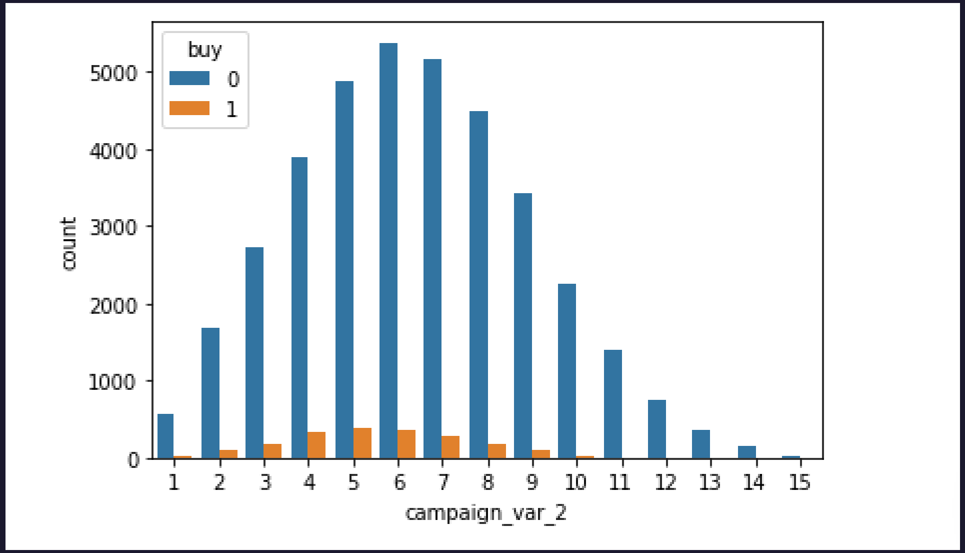
Products_purchased: let's check the distribution with respect to target



Looks like its normaly distributed if we neglect the 0 values which field by null values

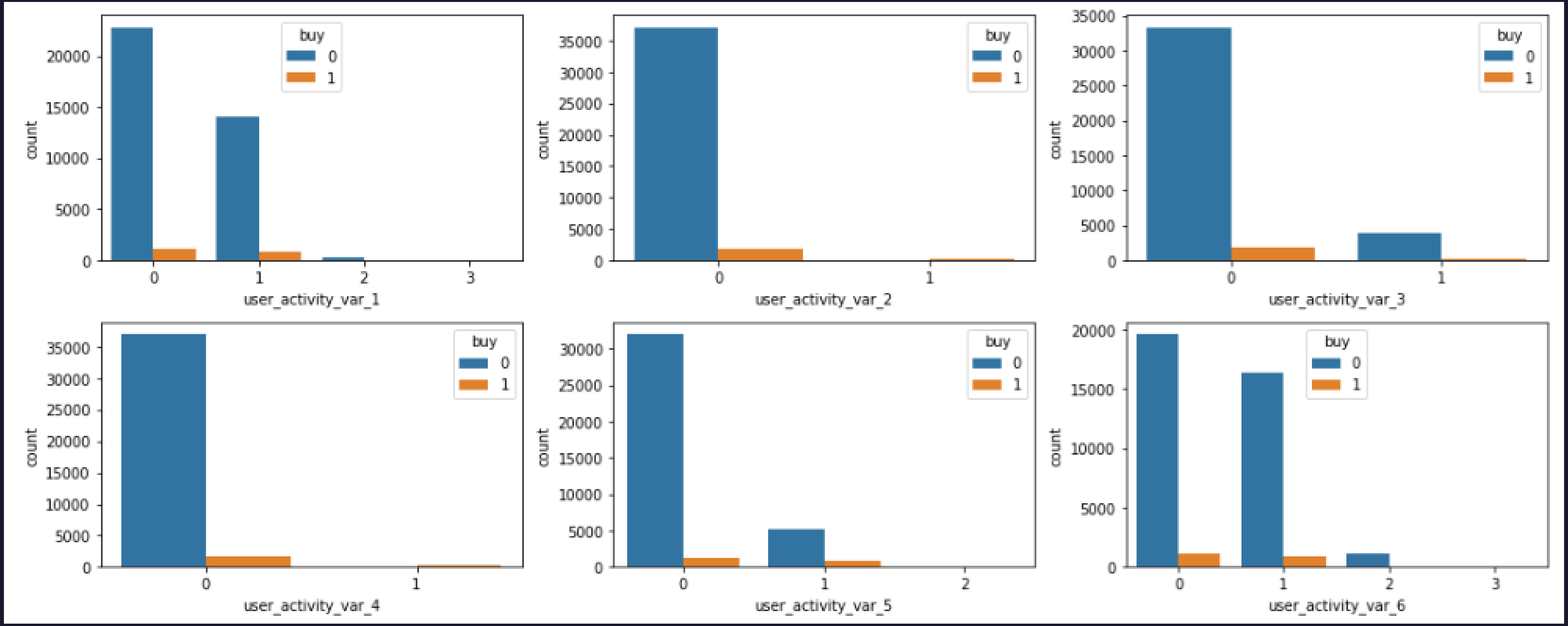
campaign_var_1 and campaign_var_2



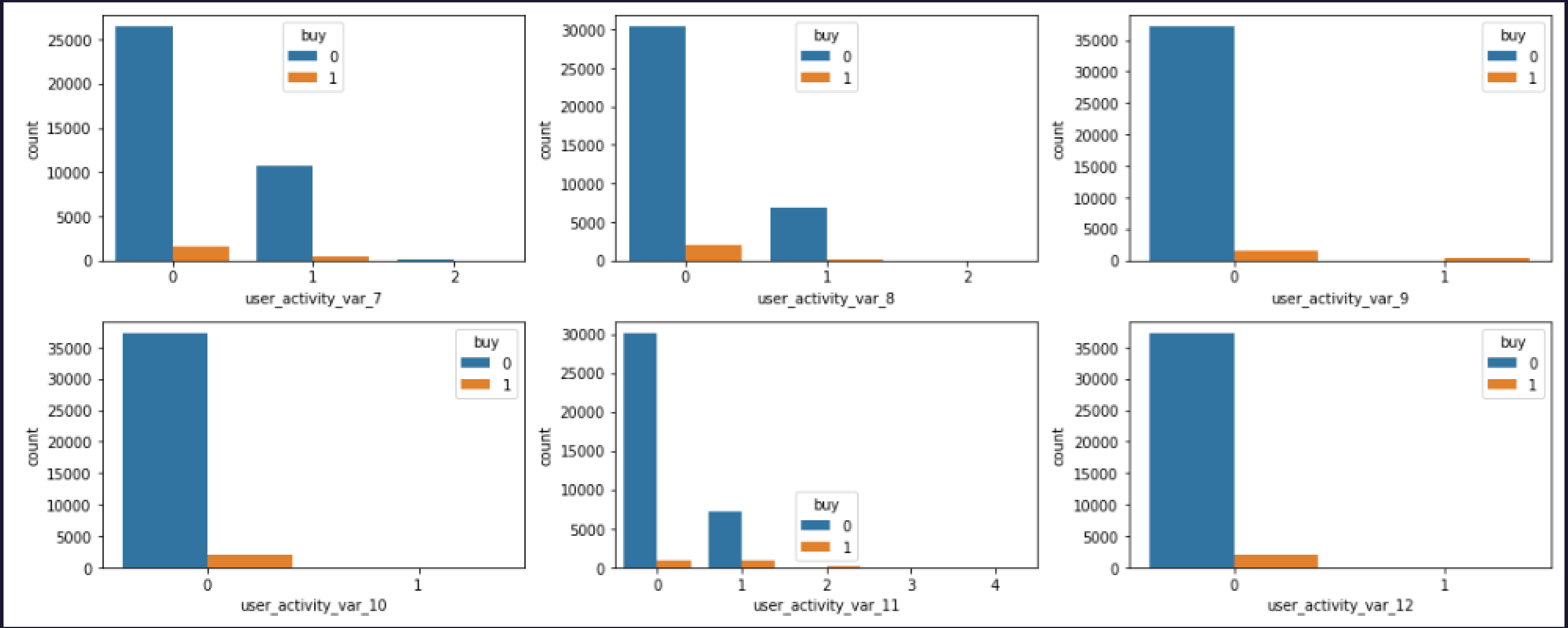


- campaign_var_1 is not normally distributed and skewed toward right direction and maximum values lie between 2 to 10
- campaign_var_2 almost normally distributed and maximum values lie between 3 to 10

Let's check the user_activity_var (1 to 12)



Let's check the user_activity_var (1 to 12)



Observation shows that some user activity has no value like
user_activity_var_10 has no values in 1
user_activity_var_11 has no values in 3 and 4
user_activity_var_12 has no values in 1
user_activity_var_5 and 6 has no values in 2 and 3

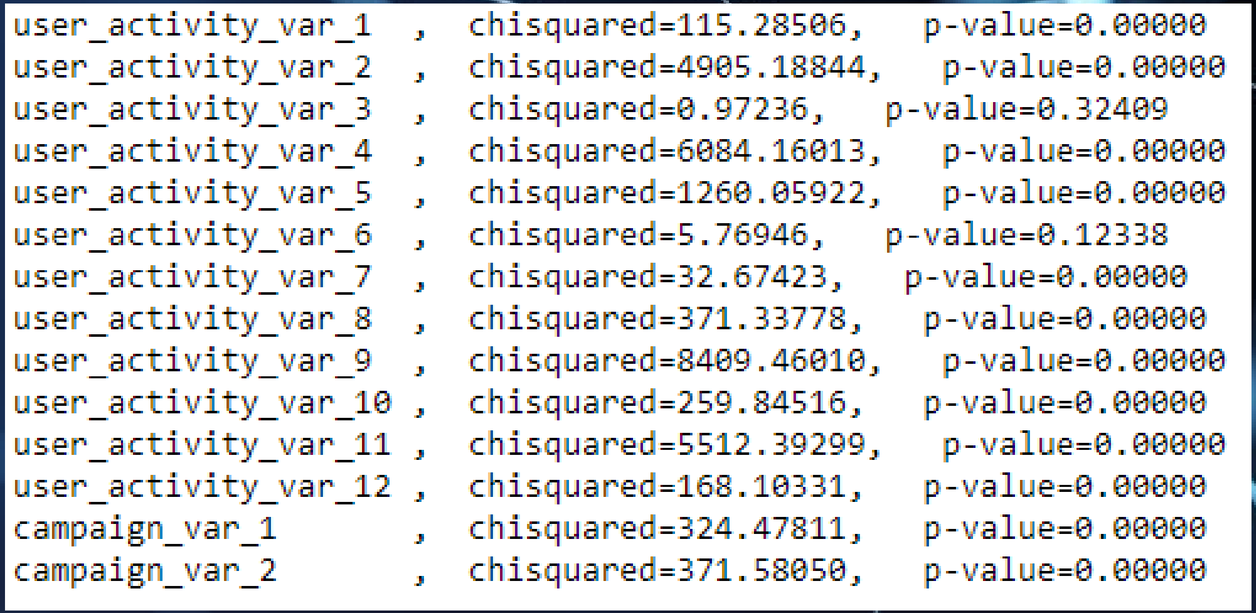
6. Statistical Significance test

As the data observered its found that the campaign_var and user_activity_var are basically the categorical variable which is encoded with numerical values so here we will do the Chi square test.

Chi square test : The chi-square test is to determine if a difference between observed data and expected data is due to chance, or if it is due to a relationship between the variables you are studying that is with target variable

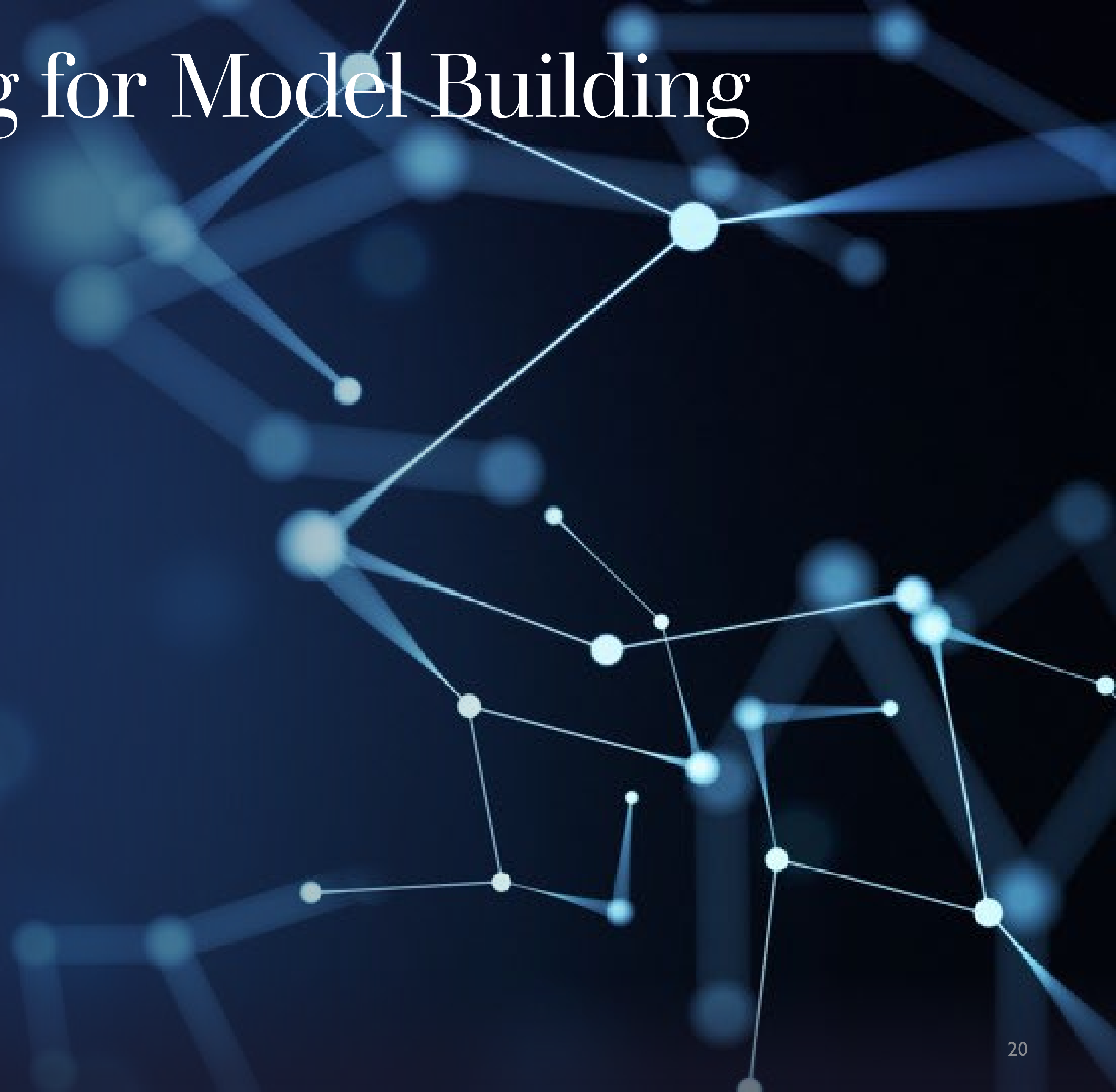
After doing the Chi square test we found that

user_activity_var_3 and user_activity_var_6 is above the significance level and we can drop this



```
user_activity_var_1  ,  chisquared=115.28506,   p-value=0.00000
user_activity_var_2  ,  chisquared=4905.18844,   p-value=0.00000
user_activity_var_3  ,  chisquared=0.97236,   p-value=0.32409
user_activity_var_4  ,  chisquared=6084.16013,   p-value=0.00000
user_activity_var_5  ,  chisquared=1260.05922,   p-value=0.00000
user_activity_var_6  ,  chisquared=5.76946,   p-value=0.12338
user_activity_var_7  ,  chisquared=32.67423,   p-value=0.00000
user_activity_var_8  ,  chisquared=371.33778,   p-value=0.00000
user_activity_var_9  ,  chisquared=8409.46010,   p-value=0.00000
user_activity_var_10 ,  chisquared=259.84516,   p-value=0.00000
user_activity_var_11 ,  chisquared=5512.39299,   p-value=0.00000
user_activity_var_12 ,  chisquared=168.10331,   p-value=0.00000
campaign_var_1       ,  chisquared=324.47811,   p-value=0.00000
campaign_var_2       ,  chisquared=371.58050,   p-value=0.00000
```

7. Data Preprocessing for Model Building



The goal of this section is to:

- ➢ Clean up columns
- ➢ Create X and y
- ➢ Split the dataset in training and test sets

From the chi square test we found that two variable is above the significance level, also the temporary variable created from Date is not too important as not found any variation In the data so dropping the same

So we are dropping below columns

'id','created_at','signup_date','month_sign','weekdays_sign','user_activity_var_3', 'user_activity_var_6', 'days_sign', 'month_create', 'weekdays_create', 'days_create'

Split the predictive variable and target variable into X and y, we will use this for our training and testing in model building

```
# Split the y variable series and x variables dataset
X = df_train.drop(['buy'],axis=1)
y = df_train['buy']
```

Fruther X and y data split into train and test, keeping 20% data for training and 80% for testing

```
# Split the dataset into the training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 42)
```

Doing random_state = 42, this help us to not see the training model, so that we can get the model performance on test data Whilch look like the unseen data

8. Model Building¶

The goal of this section is to:

- Build the multiple simple model and check the F1-macro, as here out objective is to find the potential customer who will purchase the product in next three month.
  so our macro class is One, so we will focus on f1-marco
- Classification Summary Function to check the performance of the model

Checking simple model

We are training below model, with no parameter or the minimal parameter

```
#traing simple model

models = [
    RandomForestClassifier(max_depth=5, random_state=0, n_estimators=100, n_jobs=-1),
    LogisticRegression(random_state=0),
    DecisionTreeClassifier(max_depth=5),
    GradientBoostingClassifier(),
    GaussianNB(),
    SVC(),
    XGBClassifier(eval_metric='mlogloss'),
    KNeighborsClassifier()
]
```

As we are checking the best model we have to work, we are using the cross- validation technique

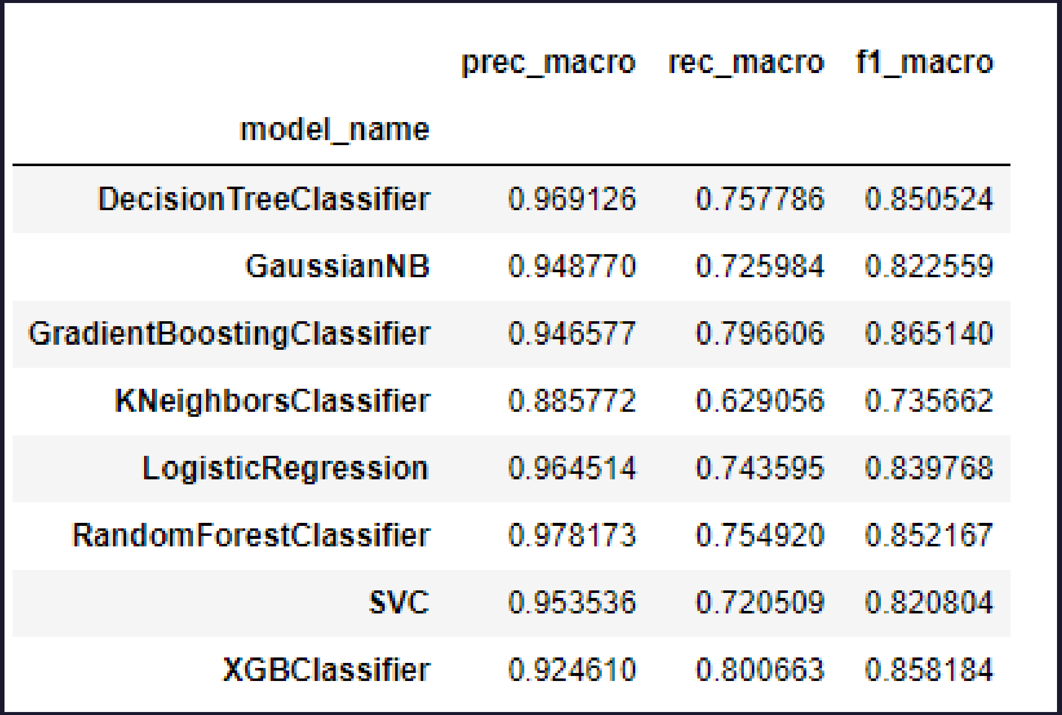
This will help to to estimate the skill of a machine learning model on unseen data

It also help to detect overfitting, ie, failing to generalize a pattern

```
#cross validation
CV=5
cv_df = pd.DataFrame(index=range(CV*len(models)))


scoring = {'accuracy' : make_scorer(accuracy_score),
           'precision' : make_scorer(precision_score),
           'recall' : make_scorer(recall_score),
           'f1_score' : make_scorer(f1_score),
           'prec_macro': 'precision_macro',
           'rec_macro': make_scorer(recall_score, average='macro')}
```

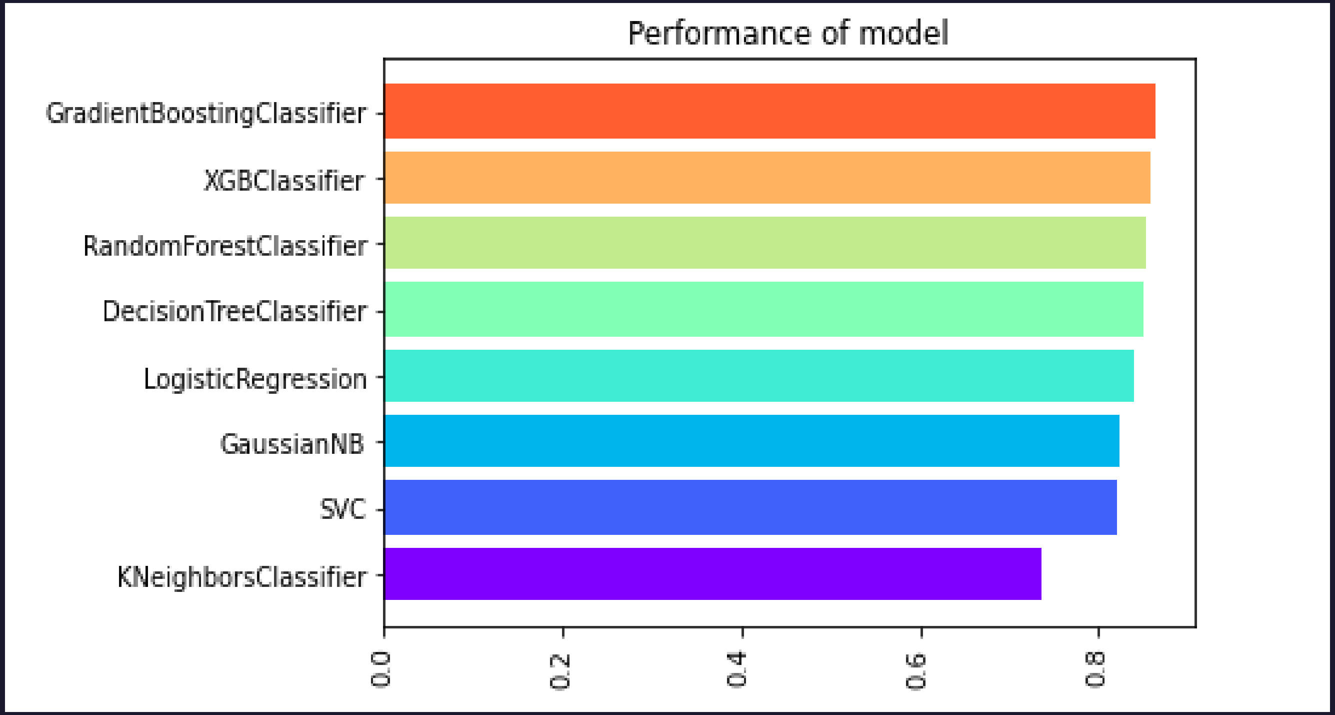
```
entries=[ ]
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_validate(model,X_train,y_train, scoring=scoring,cv=CV)
    for fold_idx, accuracy in enumerate(accuracies):
        prec_macro = accuracies['test_prec_macro'].mean()
        rec_macro = accuracies['test_rec_macro'].mean()
        f1_macro = 2*(prec_macro*rec_macro)/(prec_macro+rec_macro)
        entries.append((model_name,fold_idx,prec_macro,rec_macro,f1_macro))
cv_df = pd.DataFrame(entries,columns=['model_name','fold_idx','prec_macro','rec_macro','f1_macro’])
mean_macro = cv_df.groupby('model_name')['prec_macro','rec_macro','f1_macro'].mean()
mean_macro
```

Model performance:



model_name	prec_macro	rec_macro	f1_macro
DecisionTreeClassifier	0.969126	0.757786	0.850524
GaussianNB	0.948770	0.725984	0.822559
GradientBoostingClassifier	0.946577	0.796606	0.865140
KNeighborsClassifier	0.885772	0.629056	0.735662
LogisticRegression	0.964514	0.743595	0.839768
RandomForestClassifier	0.978173	0.754920	0.852167
SVC	0.953536	0.720509	0.820804
XGBClassifier	0.924610	0.800663	0.858184

Graphical View



Obsercation:

By seeing above model performance GradientBoostingClassifier, DecisionTreeClassifier and XGBClassifier perform very well with respect to class1

Let's do experiment on these model,

Before experimenting the model, lets build the function which will help to reduce our work

Function to Fit the Model,

```
def modelfit(model, x_data,feature,y_target,x_test,y_test_,performCV=True, printFeatureImportance=True, cv_folds=5):

    #Fit the modelorithm on the data
    model.fit(x_data[feature], y_target)

    #Predict training set:
    x_data_predictions = model.predict(x_data[feature])
    x_data_predprob = model.predict_proba(x_data[feature])[:,1]

    #Perform cross-validation:
    if performCV:
        cv_score = cross_val_score(model, x_data[feature], y_target, cv=cv_folds, scoring='roc_auc')

    #Print model report:
    print ("\nModel Report - Training")
    print ("Accuracy : %.4g" % accuracy_score(y_target.values, x_data_predictions))
    print ("AUC Score (Train): %f" % roc_auc_score(y_target, x_data_predprob))
```

```
#Predict test set:
    test_predictions = model.predict(x_test[feature])
    test_predprob = model.predict_proba(x_test[feature])[:,1]

    #Print model report:
    print ("\nModel Report Test set")
    print ("Accuracy : %.4g" % accuracy_score(y_test_.values, test_predictions))
    print ("AUC Score (Test Set): %f" % roc_auc_score(y_test_,   test_predprob))

    #print classification report
    print("\nClassification Report : \n",classification_report(y_test_, test_predictions))

    if performCV:
        print( "\nCV Score : Mean - %.7g | Std - %.7g | Min - %.7g | Max - %.7g" % (np.mean(cv_score),
np.std(cv_score),np.min(cv_score),np.max(cv_score)))

    #Print Feature Importance:
    if printFeatureImportance:
        feat_imp = pd.Series(model.feature_importances_, feature).sort_values(ascending=False)
        feat_imp.plot(kind='barh', title='Feature Importances')
        plt.xlabel('Feature Importance Score')
```

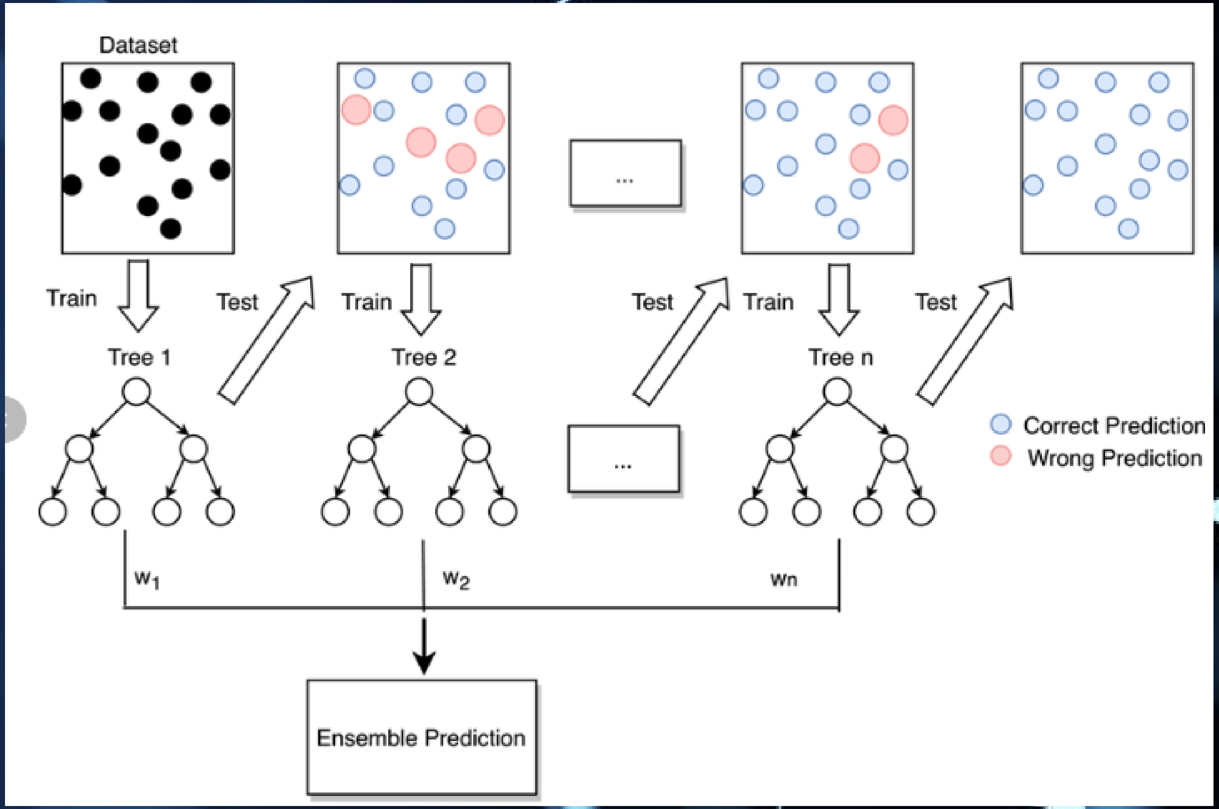
Model 1: Gradient Boosting Classifier

- Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model.
- Decision trees are usually used when doing gradient boosting
- Flow diagram of gradient boosting machine learning method.

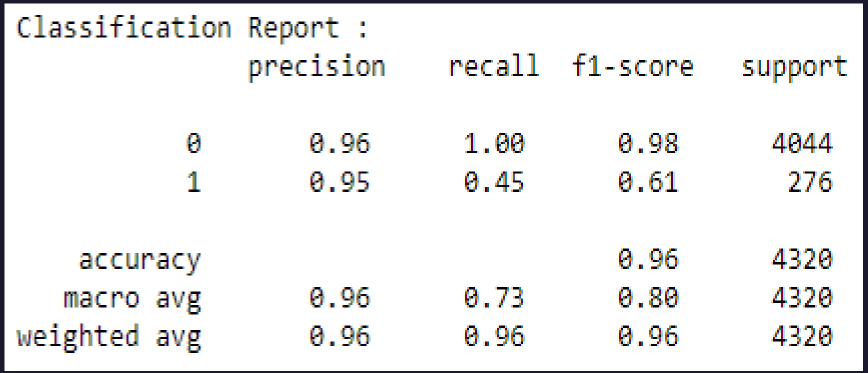
The ensemble classifiers consist of a set of weak classifiers.

The weights of the incorrectly predicted points are increased in the next classifier.

The final decision is based on the weighted average of the individual predictions



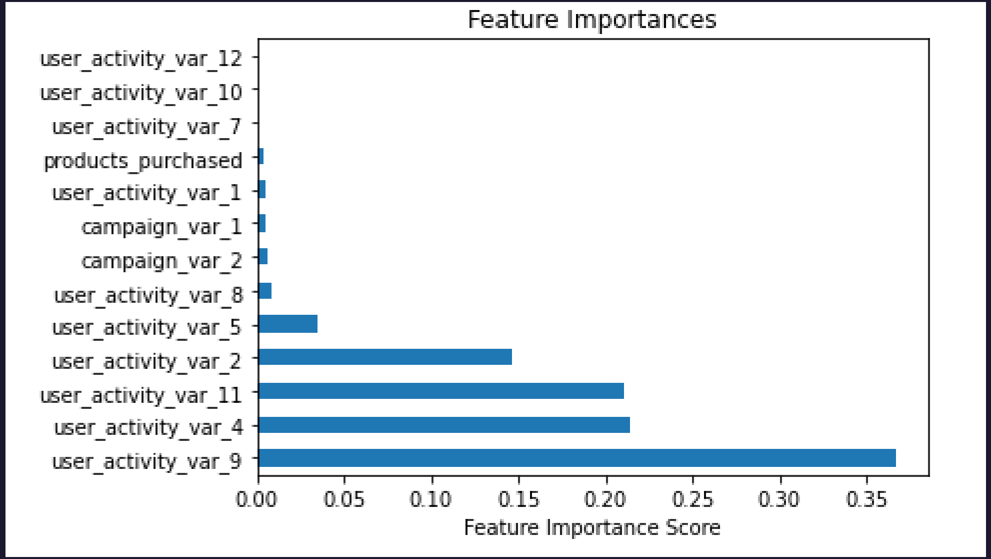
Building model with hyperparameter tunning



```
Classification Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98      4044
           1       0.95      0.45      0.61       276

    accuracy                           0.96      4320
   macro avg       0.96      0.73      0.80      4320
weighted avg       0.96      0.96      0.96      4320
```

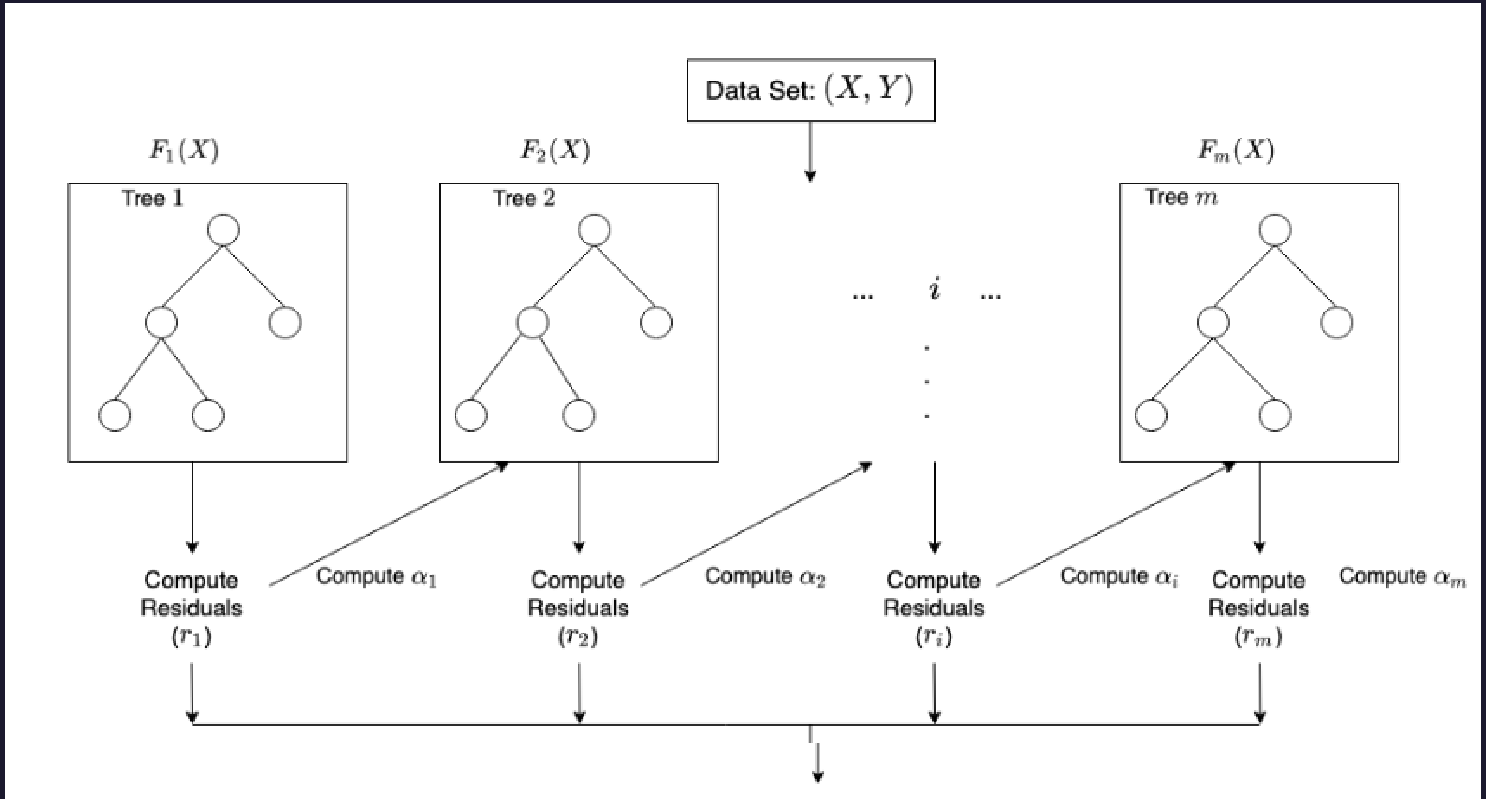
```
param = {'min_samples_split':range(1000,2100,200),
        'min_samples_leaf':range(30,60,10),
        'learning_rate':[0.01,0.05]}

gsearch = GridSearchCV(estimator =
                    GradientBoostingClassifier(learning_rate=0.1, n_estimators=60,max_depth=9,
                                max_features='sqrt', subsample=0.8, random_state=10),
                     param_grid = param, scoring='roc_auc',n_jobs=4, cv=5)

#fit model for best estimator
gsearch.fit(X_train,y_train)

#predit the model
modelfit(gsearch.best_estimator_,X_train,col,y_train,X_test,y_test)
```

After parameter tunning model is giving macro avg of .80 and also Found the some feature has no importance while predicting



Model 2. XGBoost Classifier

XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.

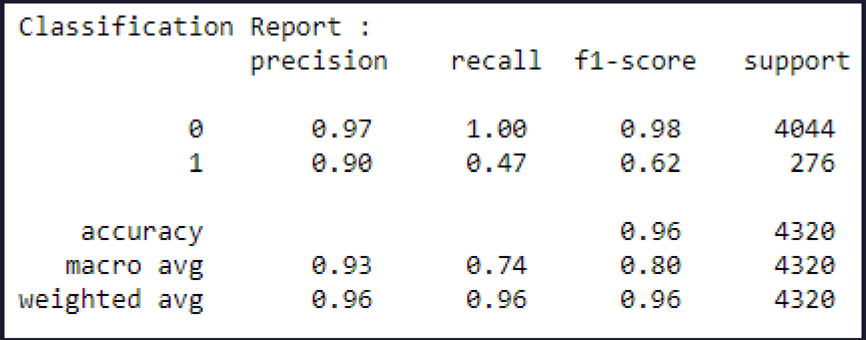


$$F_m(X) \; = \; F_{m-1}(X) + \alpha_m h_m(X, r_{m-1}),$$

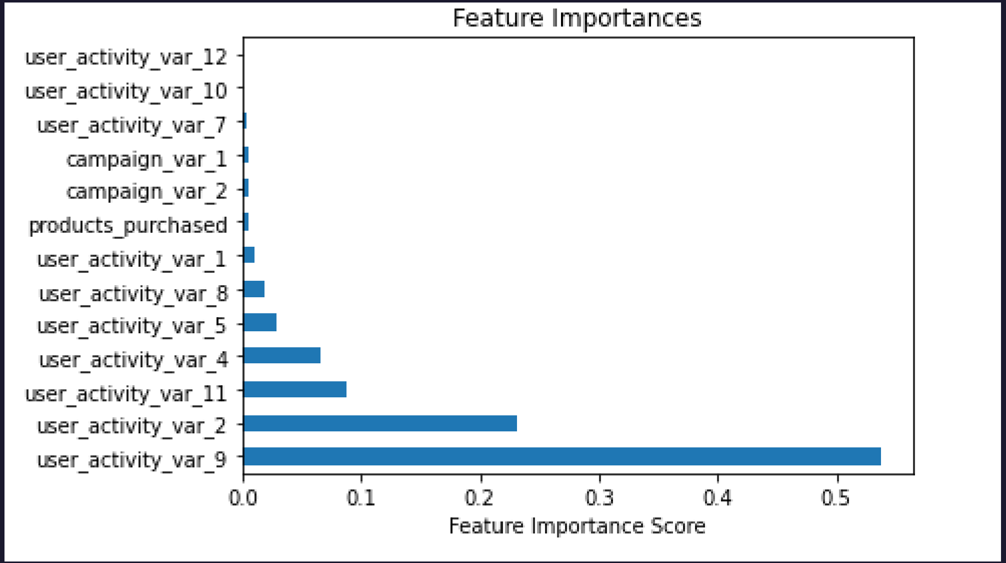
where α_i, and r_i are the regularization parameters and residuals computed with the i^{th} tree respectfully, and h_i is a function that is trained to predict residuals, r_i using X for the i^{th} tree. To compute α_i we use the residuals computed, r_i and compute the following: $arg \min_{\alpha} \; = \sum_{i=1}^{m} L(Y_i, F_{i-1}(X_i) + \alpha h_i(X_i, r_{i-1}))$ where $L(Y, F(X))$ is a differentiable loss function.

Building model with hyperparameter tunning

```
parameter = {'max_depth':[4,5,6],
           'min_child_weight':[4,5,6]
           }

xgb_grid = GridSearchCV(estimator = XGBClassifier(eval_metric='mlogloss'),
                        param_grid = parameter,
                        scoring='accuracy',
                        n_jobs=-1,
                        cv=2
                       )

xgb_grid.fit(X_train, y_train)

# fit and predit the model
modelfit(xgb_grid.best_estimator_,X_train,col,y_train,X_test,y_test)
```



```
Classification Report :
              precision    recall  f1-score   support

           0       0.97      1.00      0.98      4044
           1       0.90      0.47      0.62       276

    accuracy                           0.96      4320
   macro avg       0.93      0.74      0.80      4320
weighted avg       0.96      0.96      0.96      4320
```



9. Handling Class Imbalance

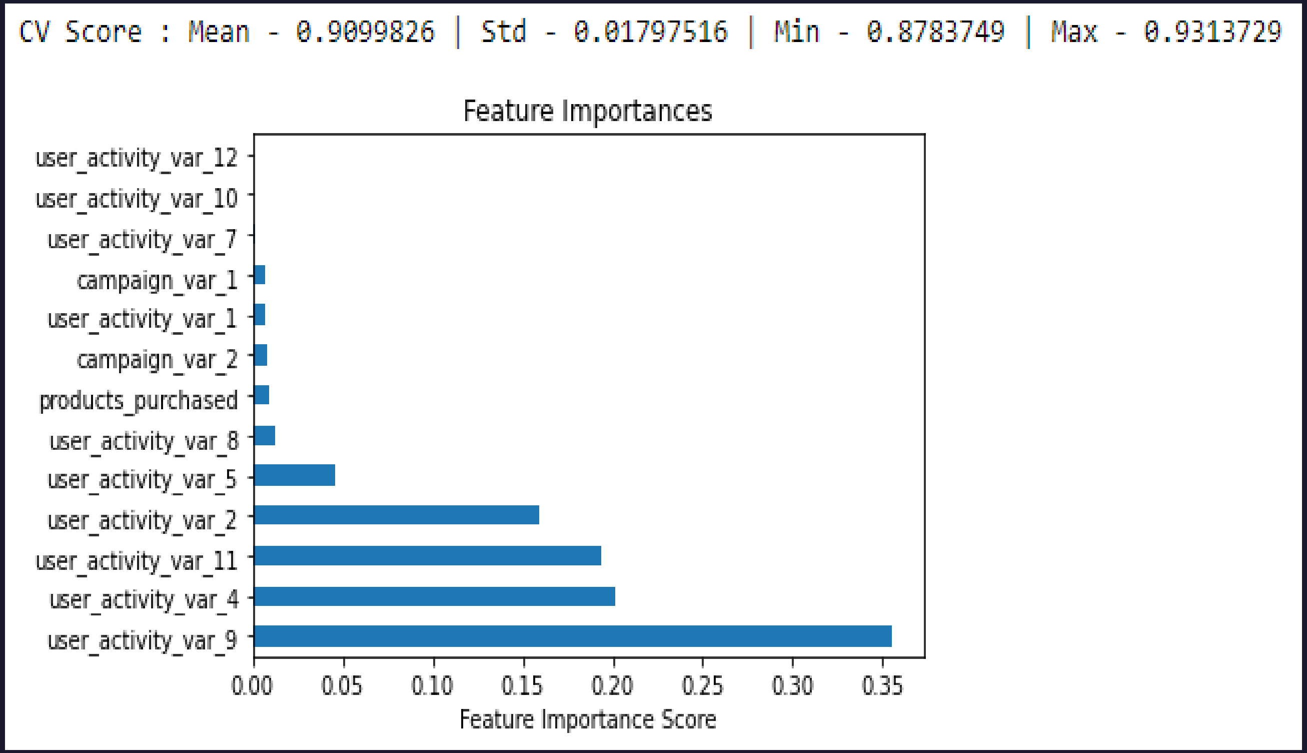
We found that the class is imbalance,

After building the model we found that the average f1-macro score is getting to 73% so trying with balance data

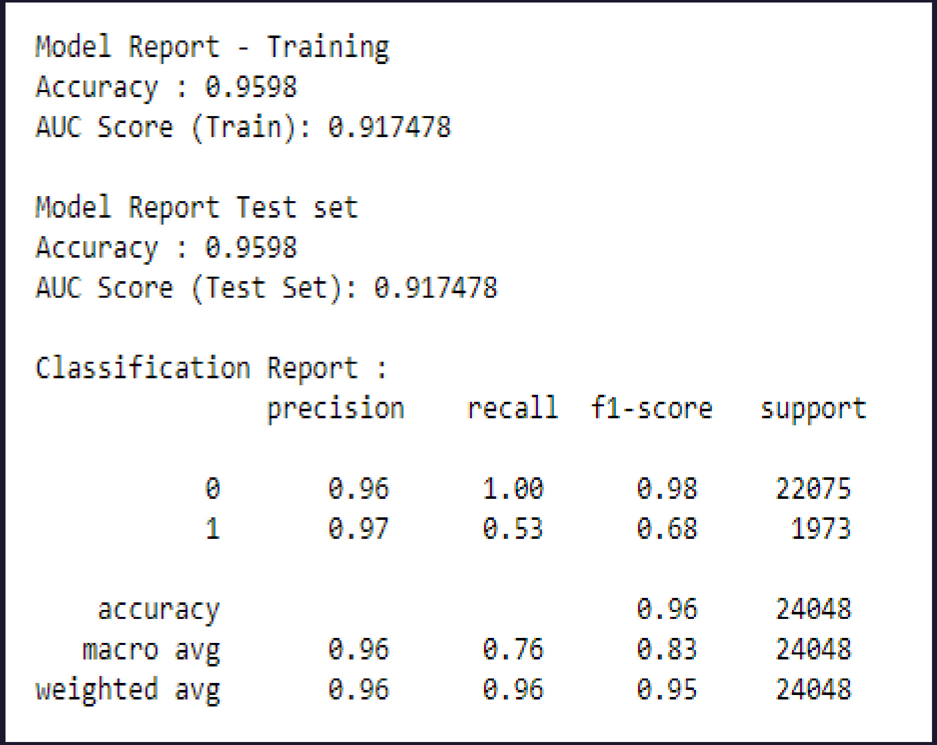
Model with down sample – here we are down sample the target variable, selecting random sample form the majority Class to match with macro class

```
#Let't try the data set with with down samping the larger class
no_buy_index = df_train[df_train['buy'] == 0].index
no_buy = len(df_train[df_train['buy'] == 0])
print(no_buy)

buy_index = df_train[df_train['buy'] == 1].index
buy = len(df_train[df_train['buy'] == 1])
print(buy)

random_indices = np.random.choice(no_buy_index, no_buy - 35000 , replace=False)     #Randomly pick up no_buy
down_sample_indices = np.concatenate([buy_index,random_indices])  # combine
down_sample = df_train.loc[down_sample_indices]  # Extract all those records to create new dataset
down_sample.shape
down_sample.groupby(["buy"]).count()  # look at the class distribution after downsample


# Split the y variable series and x variables dataset
X_bal = df_train.drop(['buy'],axis=1)
y_bal = df_train['buy']
```

Model 4: Gradient Boosting Classifier On Down Sample Mejority Class



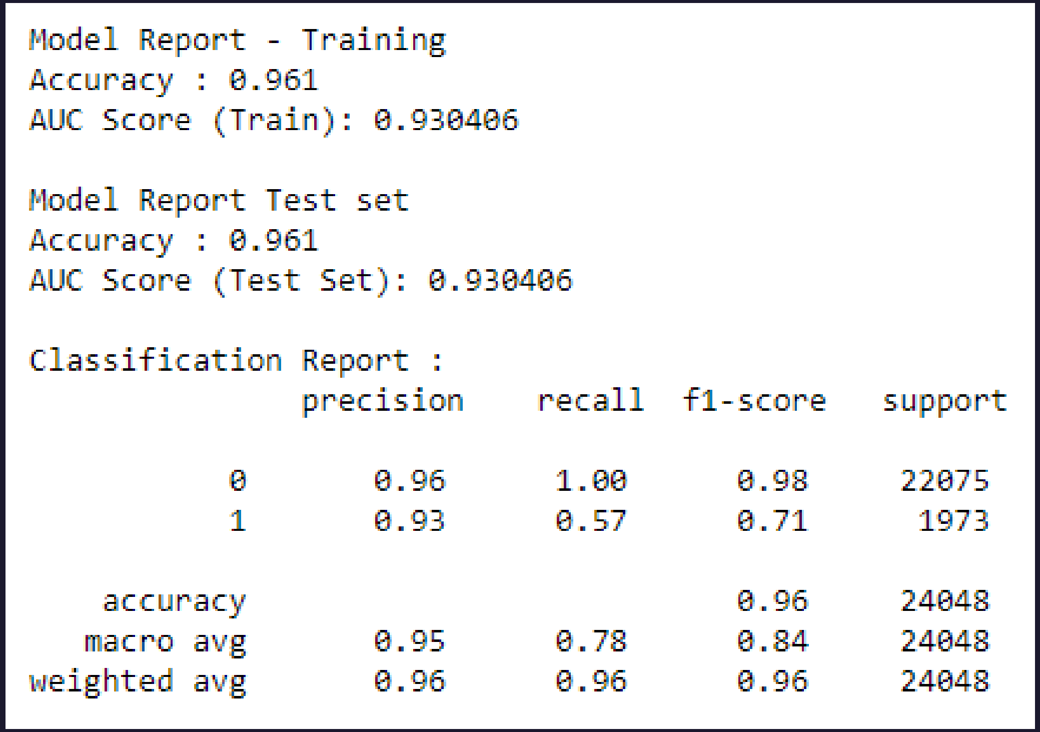
```
CV Score : Mean - 0.9099826 | Std - 0.01797516 | Min - 0.8783749 | Max - 0.9313729
```



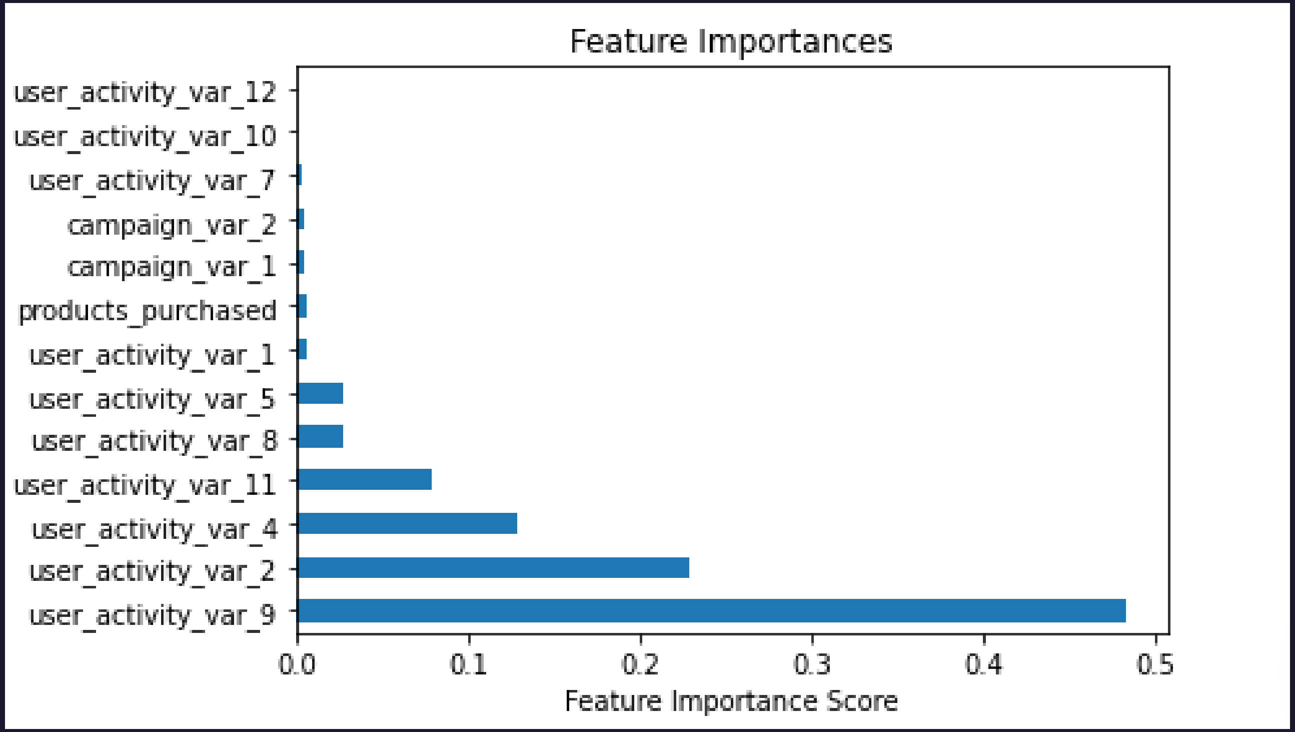
```
Model Report - Training
Accuracy : 0.9598
AUC Score (Train): 0.917478

Model Report Test set
Accuracy : 0.9598
AUC Score (Test Set): 0.917478

Classification Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     22075
           1       0.97      0.53      0.68      1973

    accuracy                           0.96     24048
   macro avg       0.96      0.76      0.83     24048
weighted avg       0.96      0.96      0.95     24048
```

Found some imprument

Model 5: XGradient Boosting Classifier On Down Sample



```
Model Report - Training
Accuracy : 0.961
AUC Score (Train): 0.930406

Model Report Test set
Accuracy : 0.961
AUC Score (Test Set): 0.930406

Classification Report :
              precision    recall  f1-score   support

           0       0.96      1.00      0.98     22075
           1       0.93      0.57      0.71      1973

    accuracy                           0.96     24048
   macro avg       0.95      0.78      0.84     24048
weighted avg       0.96      0.96      0.96     24048
```

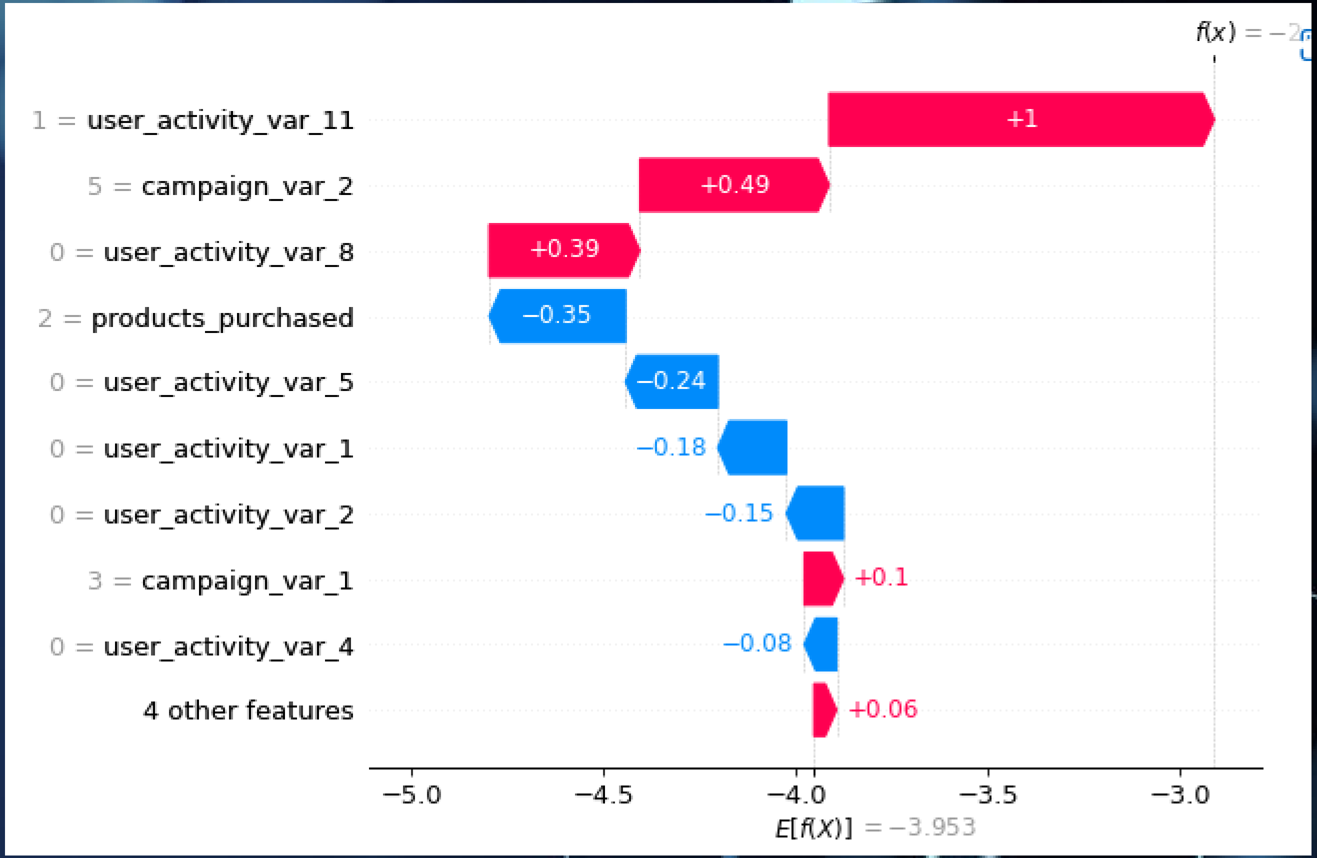


Found some imprument

11. Feature Importance & SHAP Values

From the above model and by seeing feature importance matrix we found that some feature has no role for prediction

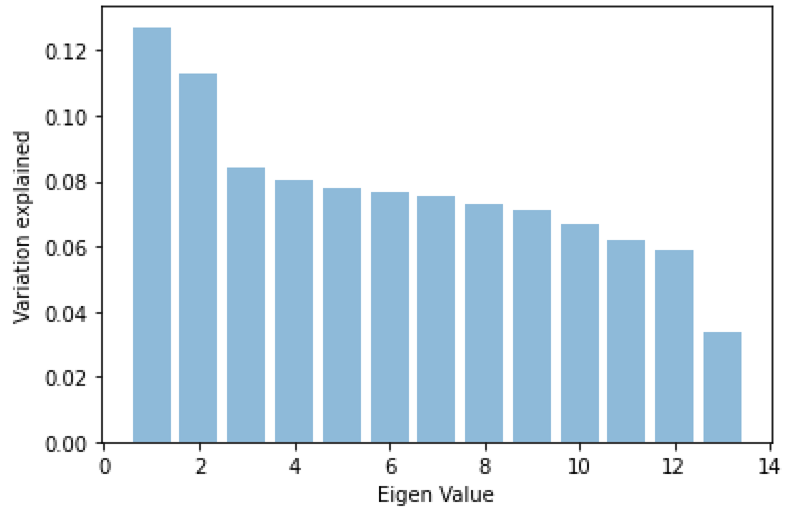
Let's do some experiment in feature selection



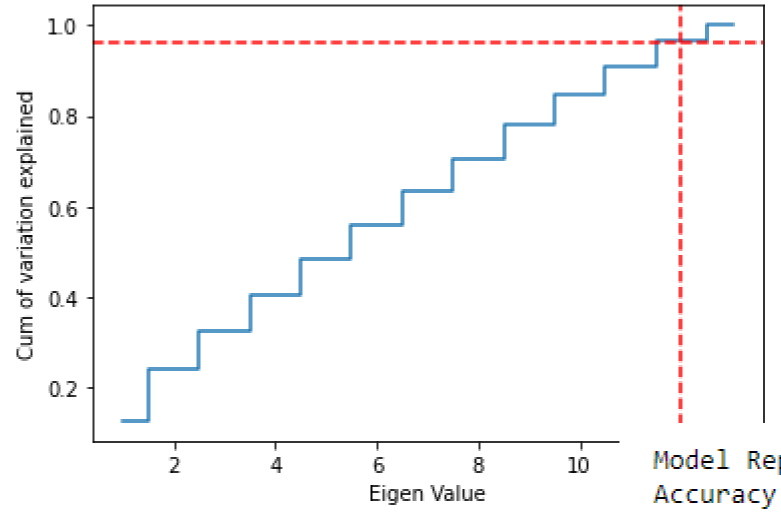
12. PCA for Dimensionality Reduction

PCA helps you interpret your data, but it will not always find the important patterns. Principal component analysis (PCA) simplifies the complexity in high-dimensional data while retaining trends and patterns

```python
plt.bar(list(range(1,14)),pca.explained_variance_ratio_,alpha=0.5, align='center')
plt.ylabel('Variation explained')
plt.xlabel('Eigen Value')
plt.show()
```

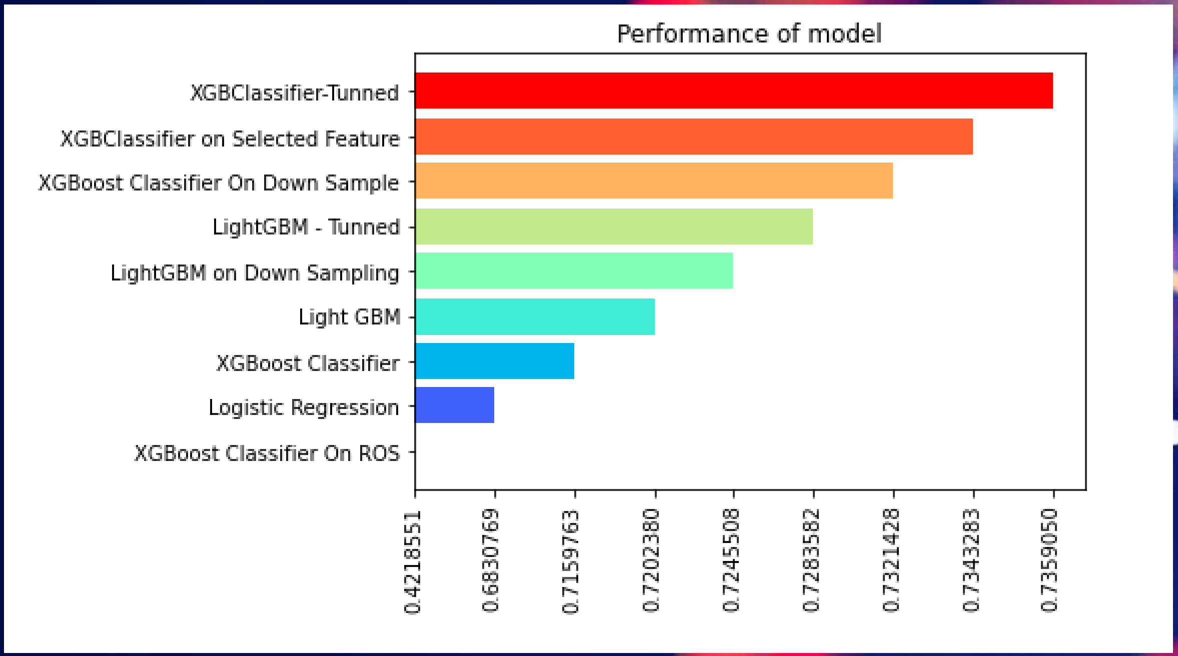


```python
plt.step(list(range(1,14)),np.cumsum(pca.explained_variance_ratio_), where='mid')
plt.ylabel('Cum of variation explained')
plt.xlabel('Eigen Value')
plt.axhline(y=0.96,color='red',linestyle="--")
plt.axvline(x=12,color='red',linestyle="--")
plt.show()
```



Not found any improvement

```
Model Report Test set
Accuracy : 0.9548
AUC Score (Test Set): 0.890845

Classification Report :
              precision    recall  f1-score   support

           0       0.96      0.99      0.98      6623
           1       0.87      0.53      0.66       592

    accuracy                           0.95      7215
   macro avg       0.91      0.76      0.82      7215
weighted avg       0.95      0.95      0.95      7215
```



Summary

After doing all the experiment, done hyperparameter tunning, balancing the data, doing feature reduction, applying pca, we found that XGB classifier do good job

So My best model is XGB Classifier with hyperparameter tunning

Thank You

ARVIND BAWANKAR

abawankar@gmail.com