Final Project

Introduction to Machine Learning

Executive Summary

The subject of this project is binary classification: predicting the chance of a user, under certain conditions, to make a purchase while browsing the site. We will use 4 models: Logistic Regression, GaussianNB, SVM, and Adaptive Boosting. In order to test the quality of the prediction, we will use the AUC measure.

First, we researched the data. In the training only, we removed observations with a high number of missing values and also outlier values in column 'B'.

For the training and test: we removed column 'D'. As to columns 'A' and 'internet browser', we grouped categories into main categories. We filled in the rest of the columns missing values by mean for 'B' (normally distributed) and by median for the other numerical columns. We created dummy values for the categorial features, removed/merged columns with correlation, normalized the data and performed PCA.

Before running the models, we performed a search for the best hyper parameters for each model that will result in the highest AUC by using k-fold cross-validation.

Then, we used the best hyper parameters for each model, created a ROC curve and built a Confusion Matrix.

We noticed that SVM provided the highest AUC for the test set (which is part of the train test we saved for evaluation). We also saw that it had the lowest train-test AUCs difference, making it the model with the least chance to overfit compared to others. For these reasons, we were convinced to choose SVM as our chosen model. We trained SVM on all training data set, found the best hyper parameters for it, ran it on the test data set and created predictions for the test file.

Exploration

At this stage, we explored the data and visualized it. We opened the train file. The train data has 10479 rows (samples) and 21 columns (features). Each sample was labeled as to whether a purchase was made (1) or not (0).

We split the data set for training and test sets - 80% for training and 20% for test. For those 2 sets, the Percentage of '0' labels is 85% while the percentage of '1' labels is 15%. That may affect the model's correctness, since it may predict label 1 less accurately because it had fewer observations labeled as '1' to practice on.

Now we will focus on the train part (80% of the training data set).

- We looked for missing values in all the columns and saw that in column "total duration" almost half of the values are missing and in column D most of the values are missing.
- We checked the types of features and saw that columns "info page duration" and "product page duration" contain the suffix "minutes". We removed that suffix andconverted it to numerical type.
- We assumed that 'Region' and 'device' are categorical features so we updated their typeaccordingly.
- We examined whether the column "total duration" (where its values are not null) is thesum of other 'duration' columns ('admin page duration', 'info page duration', 'product

page duration'). That occurs in 66% of the data. In our opinion, that percentage is not high enough. We decided not to fill in the null values of "total duration" with the sum of other 'duration' columns.

After these changes, we split the features of the data into 2 different main groups by their type: Binary and categorical columns: ['A', 'Month', 'C', 'Weekend', 'user_type', 'internet_browser', 'device', 'Region']. Numerical columns: ['num_of_admin_pages', 'admin_page_duration', 'num_of_info_pages', 'info_page_duration', 'num_of_product_pages', 'product_page_duration', 'total_duration', 'BounceRates', 'ExitRates', 'PageValues', 'closeness_to_holiday', 'B', 'D']

Analysis of Binary and categorical columns:

We created a table that summarizes each feature: the number of different values, the most common value and uniqueness of values. In addition, we visualized that group of columns by plotting and showing the count of values for each feature. We saw that some parts of the data such as 'internet browser' and 'A' have a lot of values that represent some categories like versions/sub-versions. We thought of ways to reduce these categorical values which will be discussed in the preprocessing part.

Analysis of Binary and numerical columns:

Initially, we drew a table showing the following values for each numeric feature: count, mean, std, minimum, maximum, and percentages - 25%, 50%, 75%. From this table, it can be seen that there is a difference between the std of the features, and the difference between the range of values. We thought it would be important to perform normalization.

In addition, for each feature, we presented a histogram.It can be seen that all of our features are not normally distributed, except for feature 'B'. We presented a boxplot to identify its outliers.

We created a correlation matrix and focused on medium and high correlation (above 0.55).

- "Total duration" has a high correlation with "product page duration" and "num of product pages". We may consider dropping the column total duration because earlier, we found that "total duration" has a lot of missing values.
- For each type of page we saw a correlation between the number of pages and duration. We thought of either combining these columns or to drop one of them.
- "Exit Rates" and "Bounce Rates" have a high correlation. We may consider combining these columns or to drop one of them.

We plotted samples by a number of features to detect many missing values: by observing the histogram, we saw that only a small portion of samples has less than 17 features out of 21.

Pre-processing

The preprocessing is performed equally on the training and the test sets but, in training we removed outliers from column B and we removed observations with a lot of missing values (obviously for the test, we filled in missing values, normalized and performed PCA according to the train).

Removal of observations with a lot of missing values: We removed observations that have less

than 17 features (83 samples from the train set).
Outlier removal: As we saw earlier, only column 'B' distributes normally. We identified outliers in that column using IQR, removed observations that are far more than 3 standard deviations away from the mean values (58 samples from the train set).
Removing column "D": As we saw previously, most of the values of column 'D' are missing and for that, we dropped column that column.
Changing column "internet_browser": We saw that the "internet browser" column has a lot of unique values mainly because each browser has a lot of version options. We grouped the values by the name of the browser (ignoring version) and created 4 main categories.
Changing column A: Regarding column 'A', we decided to remove the prefix "c", assuming the numeric value represents a version. That is because "A" has a lot of unique values (90). We grouped it by the number of the version and now we have 19 categories.
Filling missing values:

- Categorical Features: We decided to fill in the missing values by the most frequent value.
- Numerical Features: We decided to fill in the missing values by mean for feature B, since it distributes normally and by median for the other numerical columns.

Converting categorical features to binary: We used "One Hot Encoding" to create dummy variables.
Remove correlated features: We focused on medium and high correlation (corr>0.55).

- ❖ As we saw earlier "Total duration" is highly correlated with "product page duration" and "product page number". That's why we chose to drop the "total duration" column and also because it has a lot of missing values.
- ❖ Each type of page is highly correlated with its number of pages and its duration. We thought of dropping one of them. We randomly decided to delete the number of pages.
- ❖ "Exit Rates" and "Bounce Rates" are highly correlated. We combined these columns by averaging their values.
- ❖ Any correlations between features having the same origin column due to the creation of dummy variables (such as user type) are meaningless.
- ❖ For some correlations it is hard to conclude. For example, 'device_1' and 'internet_browser_safari' are highly correlated, but not all types of devices and specific types of browsers correlate. In these cases we decided not to change the columns.

Normalization: As we have seen in the exploration phase, the data is clearly not normalized because different features have different ranges of values. We would like to neutralize this effect by normalizing and transferring all the features to a uniform scale. It is important to normalize the data because PCA (which we would use later for dimensionality reduction) is very sensitive to data standardization. In addition, the data must be normalized since the variance of a feature is relative to the variance of the other features, and if we do not normalize the data, we might get incorrect results.

We used the StandardScaler method which gets standardized distribution with a zero mean and standard deviation of one.

Dimensionality problem:

We presented the graph of the PCA function and saw that the number of components preserving at least 99% of the variance is 43.

Initially, the dimensions of the problem were 21 different features. After removing highly correlated features, column 'D' and creating dummy variables, we got 67 different features. In conclusion, the dimensionality reduction made by PCA is from 67 to 43 Features.

Reasons why high-dimensionality may be problematic:

- Bias-Variance tradeoff: too many dimensions, higher variance.
- Similar features (by meaning) that are different by noise won't improve any model's performance but rather damage it. We better decrease the number of these features.
- The curse of dimensionality: the higher the number of features, the more our data becomes sparser, which results in overfitting.
- Longer calculation and longer running time.

Ways to identify high-dimensionality:

- PCA (as we did).
- Removal or combining features with high correlation (as we did).
- Other methods of Dimensionality reduction: Choosing the top 10 features correlated with the label, removal of features with little variation in their value. Feature selection methods such as forward and backward selection.

We gathered all the functions of the pre-processing under one function for the train and also created a pre-processing function for the test.

Model Running

In order to find the optimal model, we performed two phases: The first stage was hyper parameters selection for each model using the function GridSearchCV. That function was applied with the cross-validation strategy. We chose a 10 fold cross validation which returns the best hyper parameters for a model. The best hyper parameters are the ones with which model scored the higher AUC for the train set.

We tried out the following models: Logistic Regression, Gaussian Naive Bayes, SVM, and Adaptive Boosting after preprocessing of the train (with PCA).

Model Evaluation

In the second stage, after finding the best hyper parameters for each model, we examined the quality of the predictions by looking at both ROC plot and confusion matrix.

ROC curves include:

- Training AUC and Test AUC, after we trained the model on the training part.

- Cross-validation of the train set: We divided the train set into 10 folds. The model practiced on 9 folds, while the 10^{th} fold was saved for validation. We showed the validation AUC for each validation fold.

SVM Confusion matrix

Cells which represent the model's correct predictions:

- True Positive - 104 samples where the model predicted a purchase (1) and was right.
- True Negative - 1751 samples where the model predicted no purchase (0) and was right.

Cells which represent the model's errors:

- False Positive - 29 samples where the model predicted a purchase (1) and was wrong.
- False Negative - 212 samples where the model predicted no purchase (0) and was wrong.

We saw that the main error is where the model predicted wrongly no purchase (0). That might happen because the model was trained on data mostly labeled as '0'.

Overfitting

As we expected, there were performance gaps between training and test AUC for each model we ran. The performance gaps are not greater than 0.05. Comparing all AUC scoring gaps, we suspect that Adaboost might overfit, since its gap is relatively the highest (0.046). The other models' AUC score gaps were: 0.01,0.01,0.002.

Execution of predictions on the real test set

We saw earlier that SVM provides the highest AUC according to test data as well as the lowest AUC scoring gap - least likely to overfit. For these reasons, we chose SVM to be the model which would make the predictions on the real test set. We executed our pre-process functions on the entire training data set as well as on the real test data set. We used the best hyper parameters for SVM (found by GridSearchCV on 80% of train data because the total runtime was over the required limit). Finally, we ran the model on the real test, made predictions and saved the probabilities to purchase (label =1) for each sample in a csv file.

Summary

This project dealt with the problem of classification - binary classification according to the given information. First we explored the data, analyzed it and pre-processed it in order to maximize the AUC of any model we would choose to run. After that, We ran 4 machine learning models and found the best hyper parameters for each one of them. Finally, we submitted predictions for the real test data set by applying the SVM model (with the best hyperparameters).

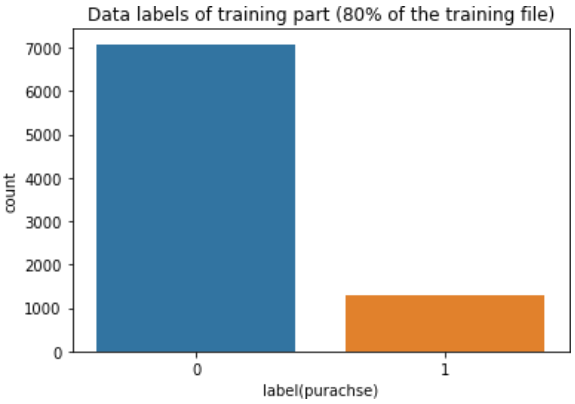
This project gave us a chance to learn a lot about the machine learning world, deal with one of its many problems. We had fun with the entire process of analyzing the data and expanding our knowledge about how a real machine learning project works.

Thank you!

Attachment 1 – Exploration

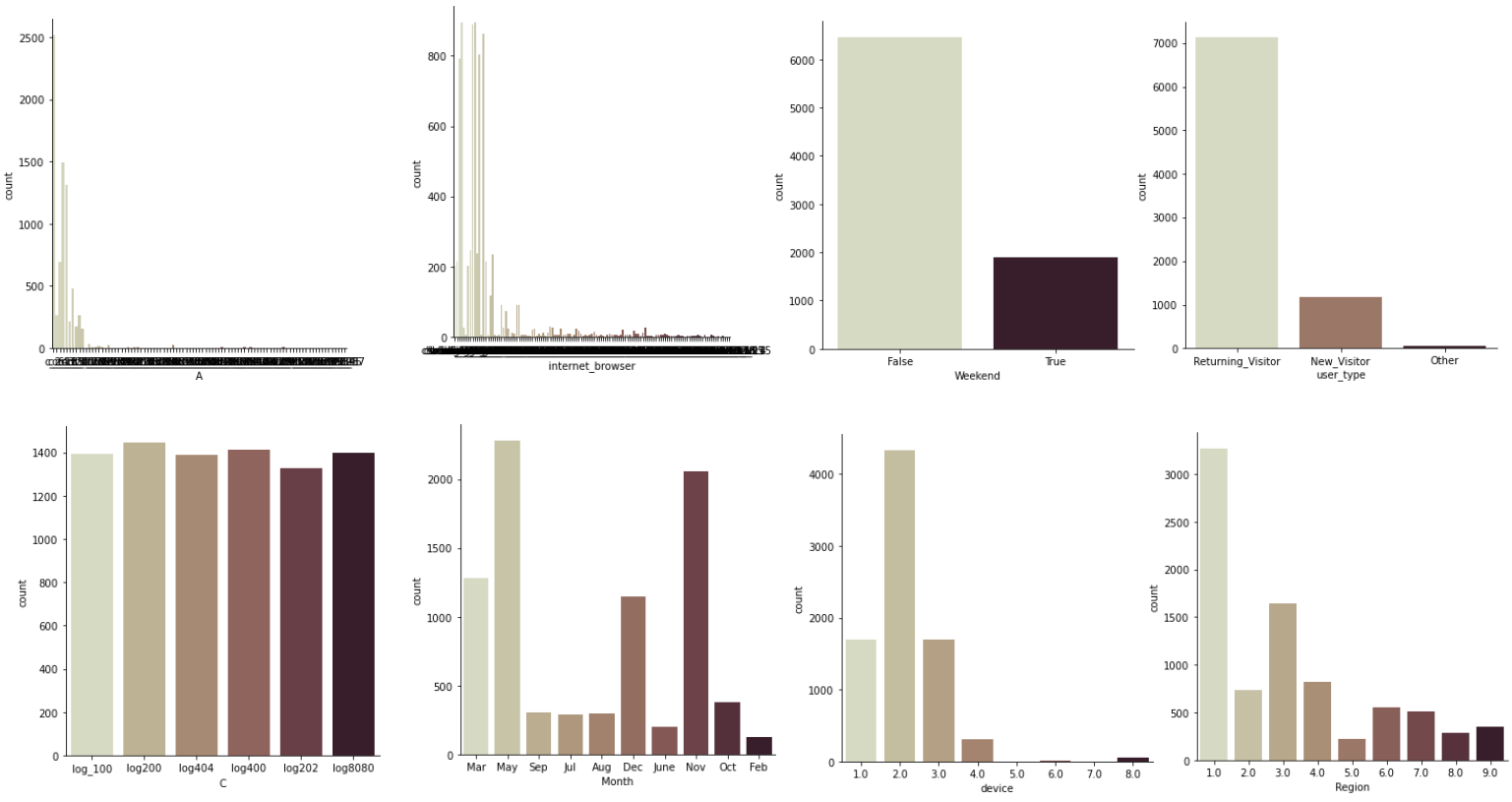
```
X.shape: (10479, 21), y.shape: (10479,)
X_train.shape: (8383, 21), y_train.shape: (8383,), X_test.shape: (2096, 2
1), y_test.shape: (2096,)
```

Attachment 1.A – train data shape

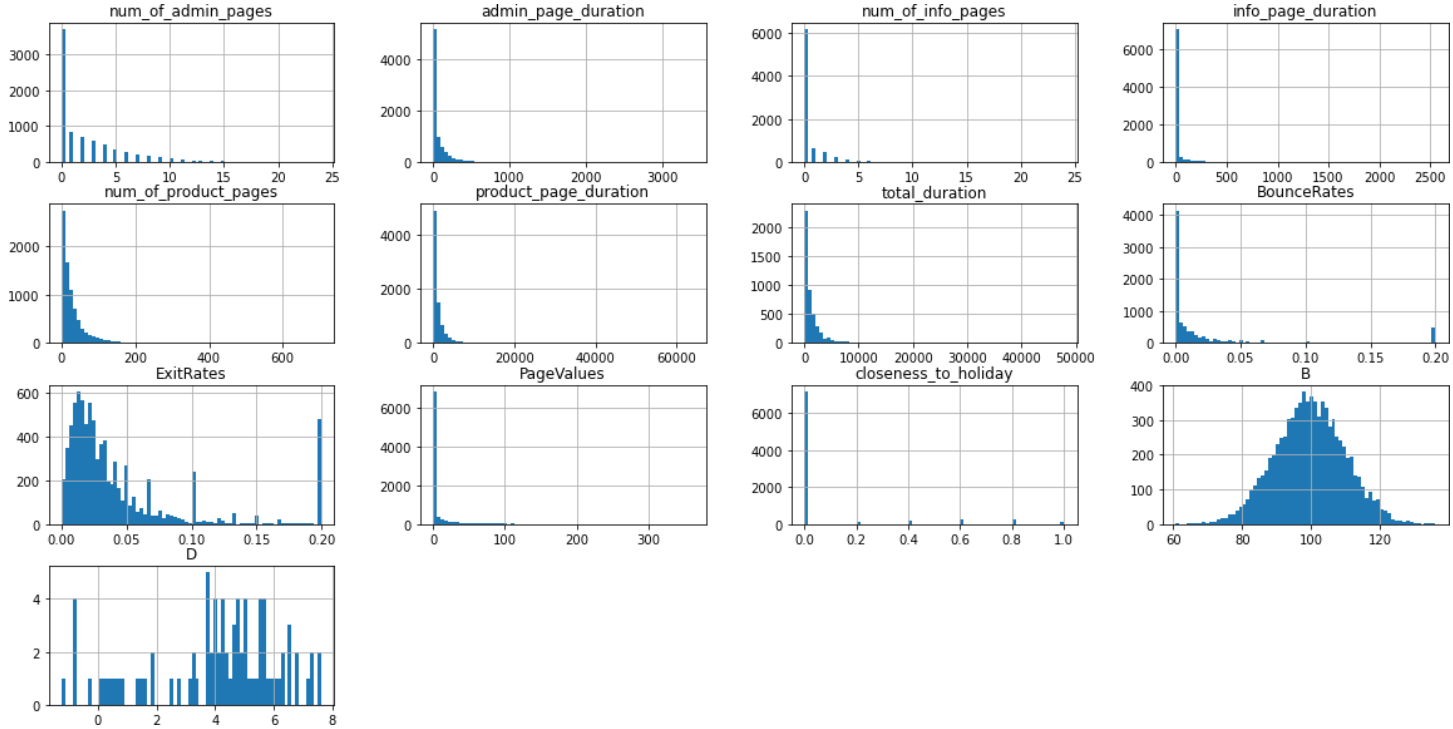


Attachment 1.B – data labels of training part

Attachment 1.C – data labels of testing part



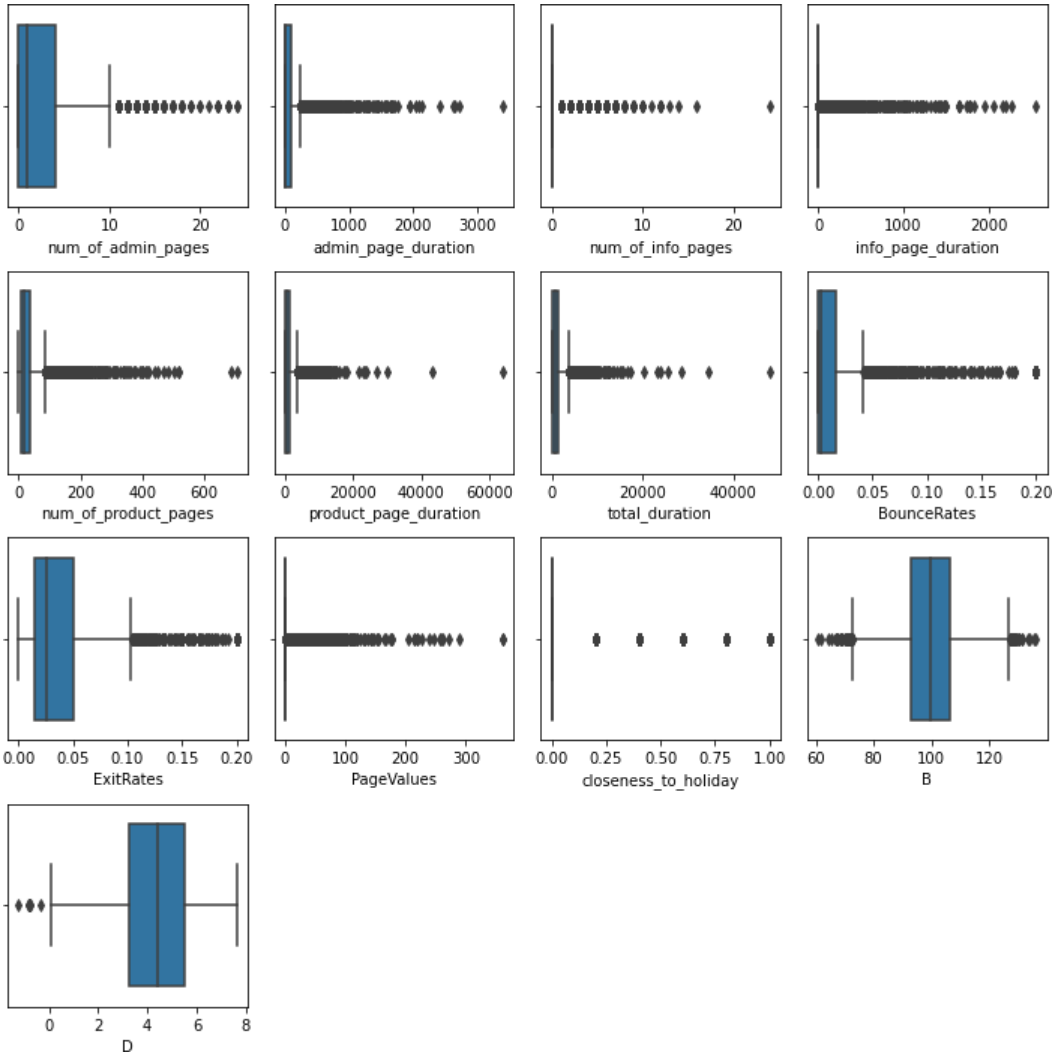
Attachment 1.D – Distribution of categorical features



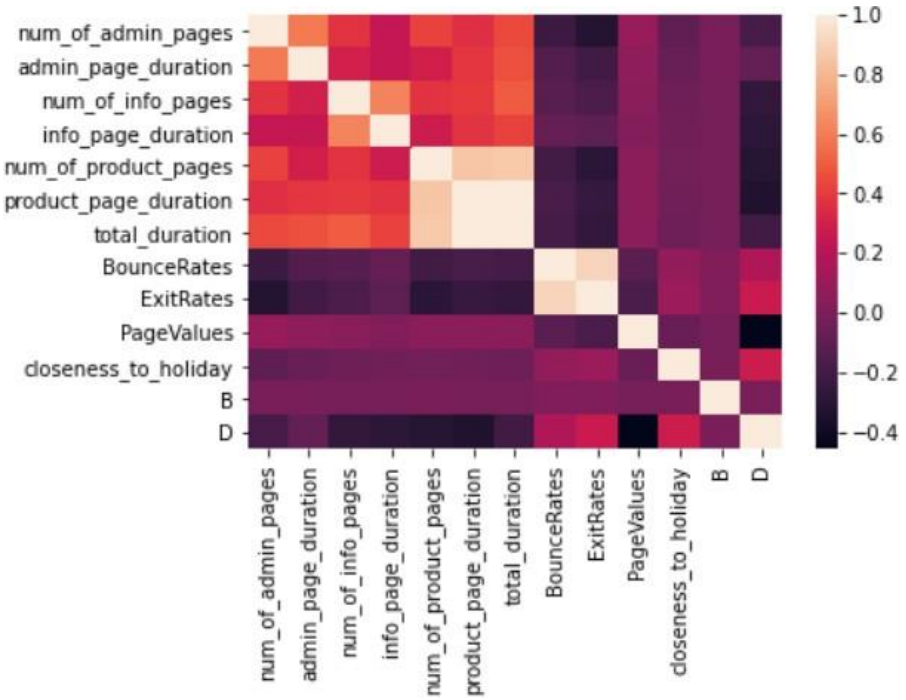
Attachment 1.E – Distribution of numerical features

```
num_of_admin_pages          472
admin_page_duration         324
num_of_info_pages           568
info_page_duration          251
num_of_product_pages        318
product_page_duration       489
total_duration             3810
BounceRates                  16
ExitRates                    22
PageValues                   18
closeness_to_holiday        396
Month                        22
device                      270
internet_browser            462
Region                       16
user_type                    17
Weekend                      15
A                           569
B                            14
C                            16
D                          8299
dtype: int64
```

Attachment 1.F – Missing values



Attachment 1.G – Boxplot of numerical features



Attachment 1.H – Correlation map of numerical features

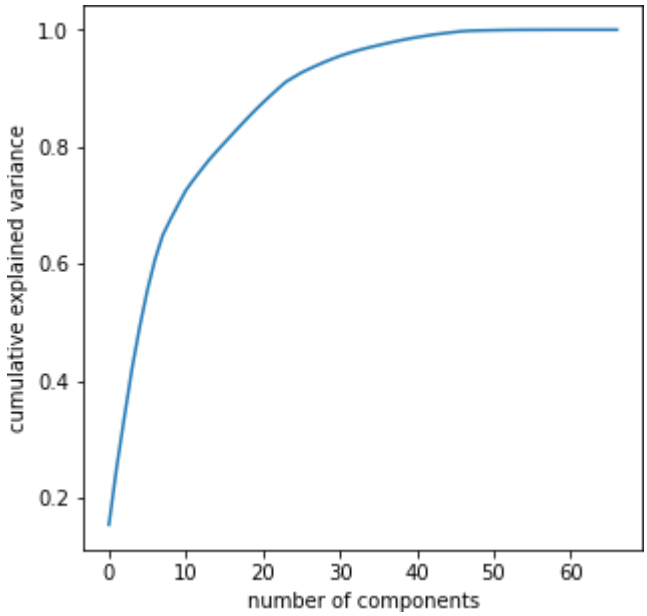
total_duration	product_page_duration	0.994679
ExitRates	BounceRates	0.912051
total_duration	num_of_product_pages	0.874135
product_page_duration	num_of_product_pages	0.855297
info_page_duration	num_of_info_pages	0.628956
admin_page_duration	num_of_admin_pages	0.589940

dtype: float64

Attachment 1.I – Correlation (> 0.55) table of numerical features

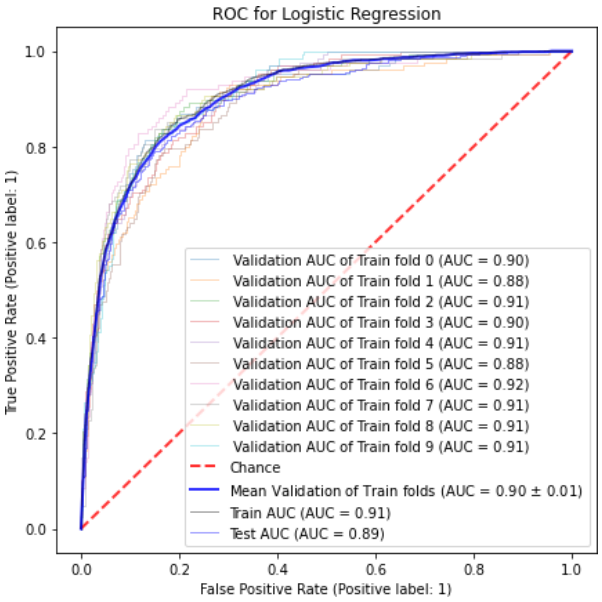
Attachment 2 – Pre-processing

```
number of components which preserve at least 99% of the variance: 43
```



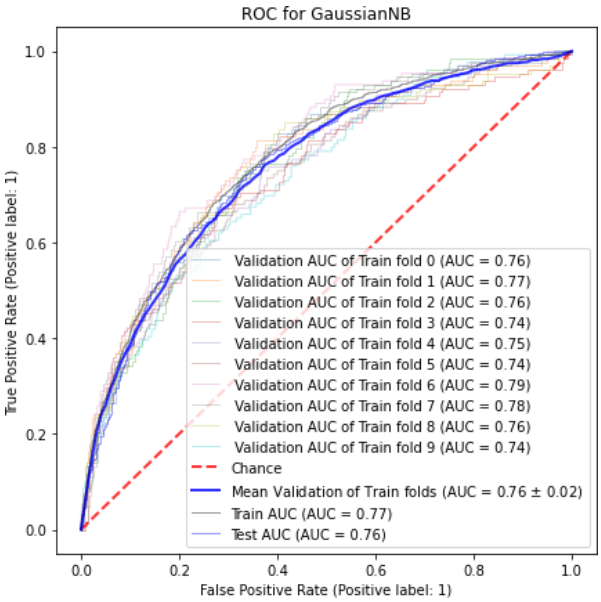
Attachment 2.A – PCA plot

Attachment 3 - Models

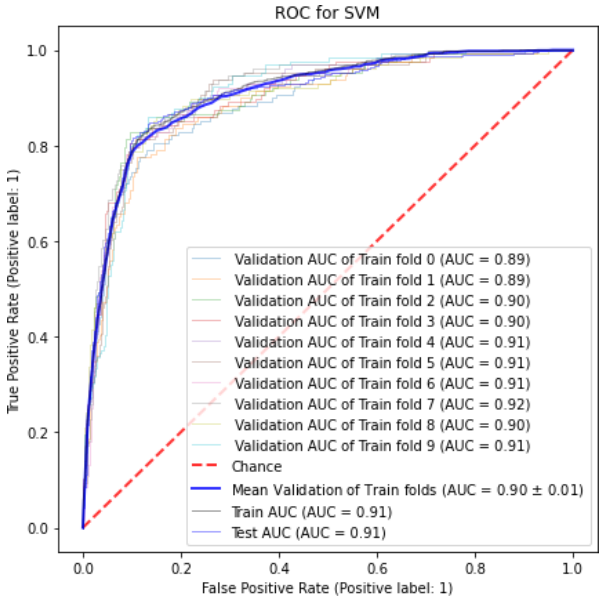



```
best hyperparameters for Logistic Regression:{'C': 0.01, 'max_iter': 50, 'penalty': 'l2', 'solver': 'liblinear'}
```

Attachment 3.A – Logistic regression ROC

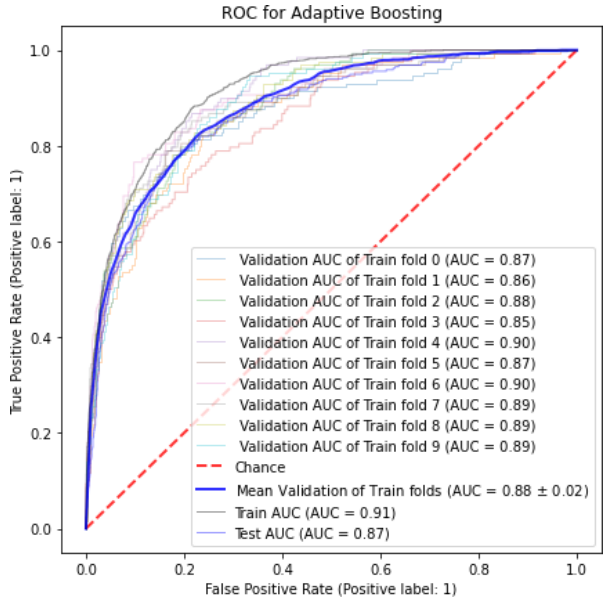


Attachment 3.B – GaussisanNB ROC



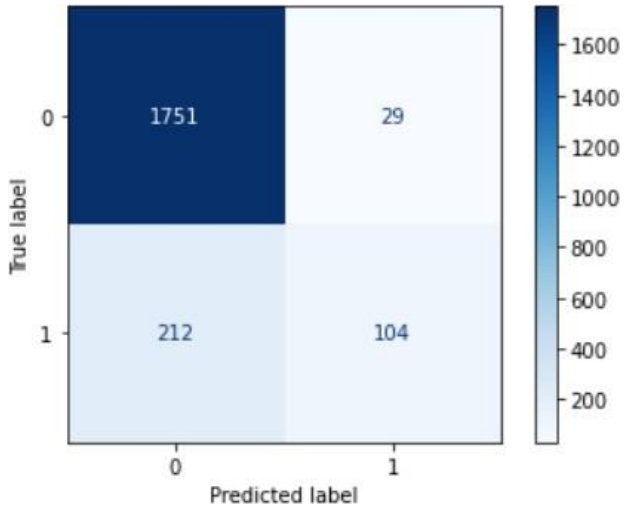

```
best hyperparameters for SVM:{'C': 0.1, 'gamma': 0.01, 'kernel': 'rbf'}
```

Attachment 3.C – SVM ROC




```
best hyperparameters for Adaptive Boosting:{'n_estimators': 50, 'random_state': 0}
```

Attachment 3.D – Logistic regression ROC



Attachment 3.E– confusion matrix of SVM model

Score Type	Logistic Regression	GaussianNB	SVM	Adaptive Boosting
Train AUC	0.906141	0.773759	0.908732	0.913609
Test_AUC	0.893045	0.763117	0.906324	0.867357
Difference	0.013096	0.010642	0.002409	0.046252

Attachment 3.H– AUC table