Machine Learning - Final Project

Introduction:

The project is about online purchases, based on a given data set ("train") with 10,479 records and 22 features, while 4 of them are anonymous. We want to train a model, using methods learned in class, that will help us to predict the test dataset sessions, whether they ended with purchase or not.

We are facing a binary classification problem, whereas
purchase = 1 , and no purchase = 0 .

Part 1 & Part 2: Exploration and Pre – Processing :

We chose to combine between these two parts, since there is straight and continuous connection between them.
we started observing the given train data set (train.csv).
We used a several functions that helped us research the file from an initial look, observing the features and their representation (we have 8 non numerical features, 14 numerical and 1 label "purchase").

We've noticed that there are a few features that aren't representing their correct attribute. For example : "info page duration" shown as a string but represents a number. The opposite : "device" and "Region" has objects attributes but shown as numerical.

We checked how balanced the train data labels, to see the proportions between the sessions, and found out that 85% of the sessions didn't end with a purchase while only 15% did. The composition of the data matters since in order to predict the test well, a large imbalance will affect the type of observations the model will practice on.

Missing values : Handling missing values is very important, missing values are not indicate the quality of the data and some machine learning algorithms don't support missing values. So we defined functions that we'll use to fill the missing values. (with median, mean or most frequent value, depending on the feature).

WEEDEND & USER TYPE: used to be object types, we changed them to 0 or 1 (Boolean). The user type feature had also "other" category, and we assumed that it's also made by a new visitor. We filled their missing values with the median since they are Boolean.

INTERENT BROWSER : we saw in the visualization that it's divided to multiple browsers (126 types) which have 4 types in common, we split those browsers into 4 groups and made them "dummy features". Since it's a categorical feature, we fill its missing values with the most frequent value.

We recognize that those 3 features: "info_page_duration", "admin_page_duration" , "product_page_duration" , have a similar session information, so we filled their missing values with the median and merge them into one feature called "total_pageDuration". And the same for these 3 features: "num_of_info_pages" , "num_of_admin_pages" , "num_of_product_pages" , merged into one feature called "total_pageNumbers".

"total duration" -> "total session duration" : we changed its name to emphasized the feature information. After the correlation matrix, we found that this feature has a very high correlation with "total page duration " (97%), and a higher correlation with "total page numbers", also we saw that it has a higher amount of missing values, therefore we decided to delete it.

MONTH & DEVICE & REGION: categorical attributes, we filled their missing values with the most frequent value and change them to dummy features.

EXIT RATE & PAGE VALUES : represent similar &BOUNCE RATE information, in the visualization we saw that they have a right skewed distribution, therefore we filled their missing values with the median value.

B: anonymous column, with a normal distribution, so we filled its missing values with the mean value.

C : anonymous column with categorial attributes, which seems like it represents the logging time of the user. We chose to delete its missing values, since it has a minor amount (only 16), then we change it to dummy feature.

A : anonymous column with a lot of missing values (700). After considering and after the forward selection (feature selection) , we decided to delete the column.

D : anonymous column with categorial attributes, it has a lot of missing values, after filling all the other missing values and creating the dummy variables, we decided to fill it's missing values with KNN imputer. When we tried removing this feature our models got a higher AUC score- therefore we decided to delete it.

ID : identification of the customers that did each session.

Purchase : The data's label we need to predict on in the test.

We used one function that grouped all these functions together, later we'll use it on the test dataset.

Normaiztion: from what we learned in class and by searching, we wanted to find the best way to normalize our data, we saw that the "sqrt method" which is great for skewed distribution as we saw in the visualization, and we visualized that normalization on our data. We then normalized our data once in a min-max scaled (which will create a single scale between -1 to 1) and that way will allow us to reduce dimensions correctly in our data. And also, we copied our data and did standard normalization – this will make all the means to 0 and the standard deviation to 1. That way we can have 2 data's we can run our models on and choose the one with higher AUC score.

Dimension reduction -Forward Selection: reducing the dimensionality of data by selecting only a subset of measured features (predictor variables) to create a model. Feature selection algorithms search for a subset of predictors that optimally models measured responses, subject to constraints such as required or excluded features and the size of the subset. The main benefits of feature selection are to improve prediction performance, provide faster and more cost-effective predictors, and provide a better understanding of the data generation process. The training time of the model or its architectural complexity may cause the model to overfit. If the model trains for too long on the training data or is too complex, it learns the noise or irrelevant information within the dataset. And so, by reducing dimensions we can help to prevent that.

Choosing the hyper-parameters for each model: we used "grid search" function - a tuning technique that attempts to compute the optimum values of hyperparameters. performed on specific parameter values of each model and doing all the possible combinations between the features to find the best ones for each model.

Models: When training a model, we must split the data in order to prevent over-fitting and also the validation process gives information that helps us tune the model hyperparameters and configurations accordingly, then we can perform it on the test dataset. When model tries to fit the entire training data , it memorized the data patterns and can cause over-fitting.

Simple models (KNN & Logistic Regression):

We used both normalization methods (standard and min-max scale) on the simple models and saw that they both had higher results with the standard normalized data, so we decided to continue modeling only with standard normalization.

KNN: stores all the available data and classifies a new data point based on the similarity (decided by the k nearest neighbors that were choosen by grid search).

Logistic regression: estimates the probability of an event occurring, based on a given dataset of independent variables.

Advanced Models (Random Forest & MLP):

Random Forest: combines the output of multiple decision trees to reach a single result. Its ease of use and flexibility have fueled its adoption, as it handles both classification and regression problems.

MLP: as we learned in class, multilayer perceptron (MLP) is a feedforward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. (Attached is a picture that illustrates the way of calculation).

Evaluating:

to evaluate the model that had the highest AUC, we used confusion matrix and K-FOLD validation. Confusion Matrix is illustrating the prediction of the model and showing whether the model had predict correctly or not, both the YES and NO labels. K-FOLD validation used to test the effectiveness of machine learning models. it shows the accuracy of the predictions that our model is putting out by a re-sampling procedure. To perform it, we need to keep aside a sample/portion of the data on which is not used to train the model, later use this sample for testing/validation. Splitting the data can affect dramatically the model ability to predict well.

Over fitting & Under fitting:

Overfitting- good performance on the training data, poor generalization to other data, underfitting- poor performance on the training data and poor generalization to other data. We have addressed this part by using the learning curve of our train and validation parts of the data. (Unfortunately, the plot showed a model that is not well fitted).

We believe that the plot shows a consequence of the multiclass data we're using. With fewer training examples, having fewer numbers of images of each class (because one of the lines tries to keep the same class distribution in each fold of the cv), so with regularization (if you call C=1 regularization, that is), it may be harder for our model to accurately guess some of the classes.

Predicting the test:

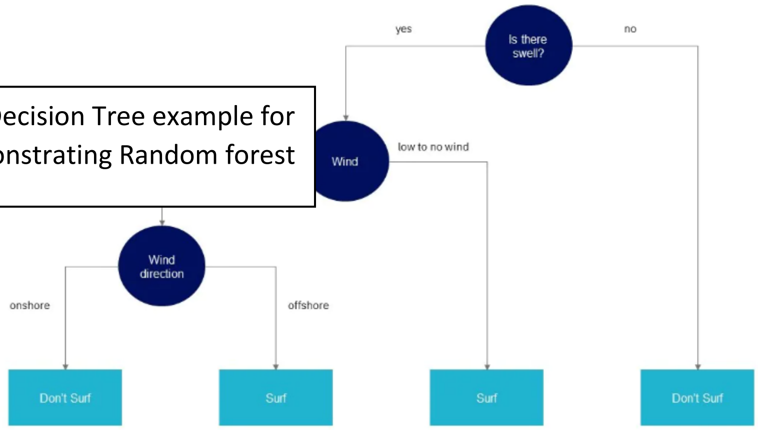
we ran the model with the highest AUC score with the parameters the Grid Search function found, only this time we didn't split the train data and didn't normalize it, then we trained our model (using the fit function) and ran our model on the test and predicted the label. We did it twice- once using forward selection and once without.

Without forward selection- we found out that the predictions are rounder (The train data had more features and therefore a larger pattern to follow therefore it was a clearer classifying the predictions.

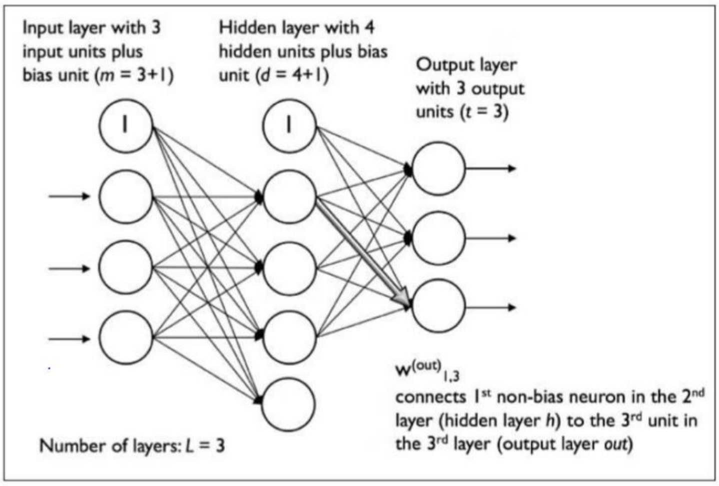
With forward selection- the predictions were more accurate since there were much fewer features, and the model was based more about certain values to determine whether the purchase happened or not.

Appendices

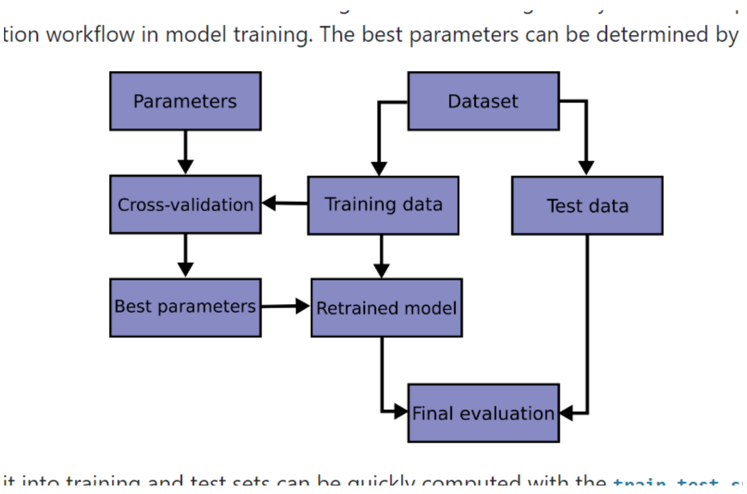
Decision Tree example for demonstrating Random forest



Multi-layer example for demonstrating how it calculated

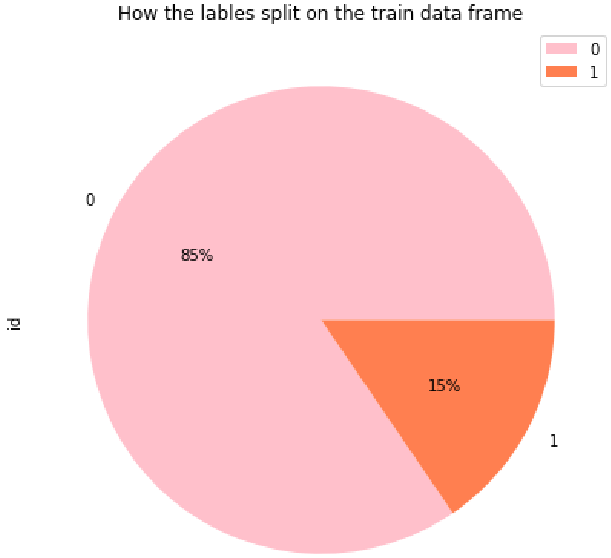

Input layer with 3 input units plus bias unit (m = 3+1)
Hidden layer with 4 hidden units plus bias unit (d = 4+1)
Output layer with 3 output units (t = 3)
1
1
$w^{(out)}_{1,3}$
connects 1st non-bias neuron in the 2nd layer (hidden layer h) to the 3rd unit in the 3rd layer (output layer out)
Number of layers: L = 3

K FOLD demonstration

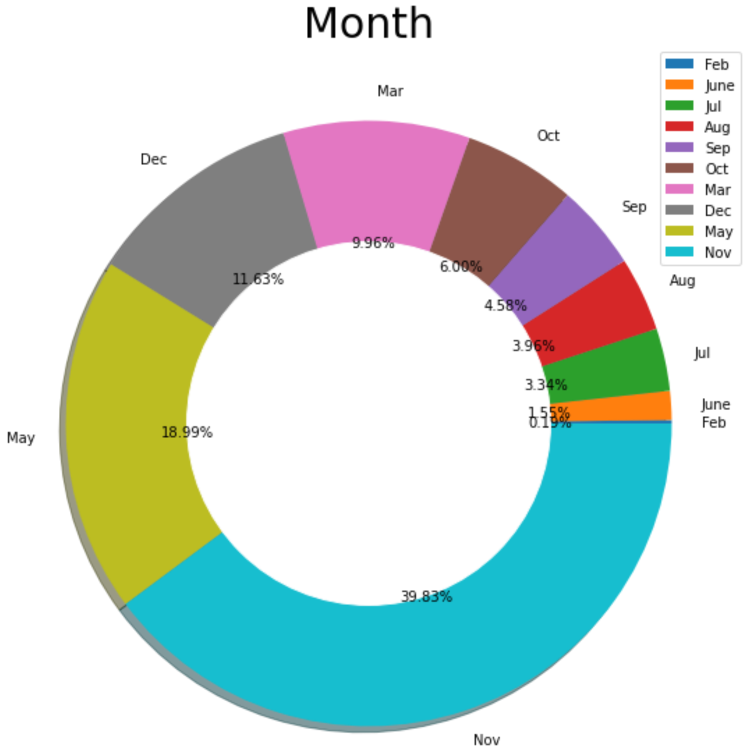

tion workflow in model training. The best parameters can be determined by

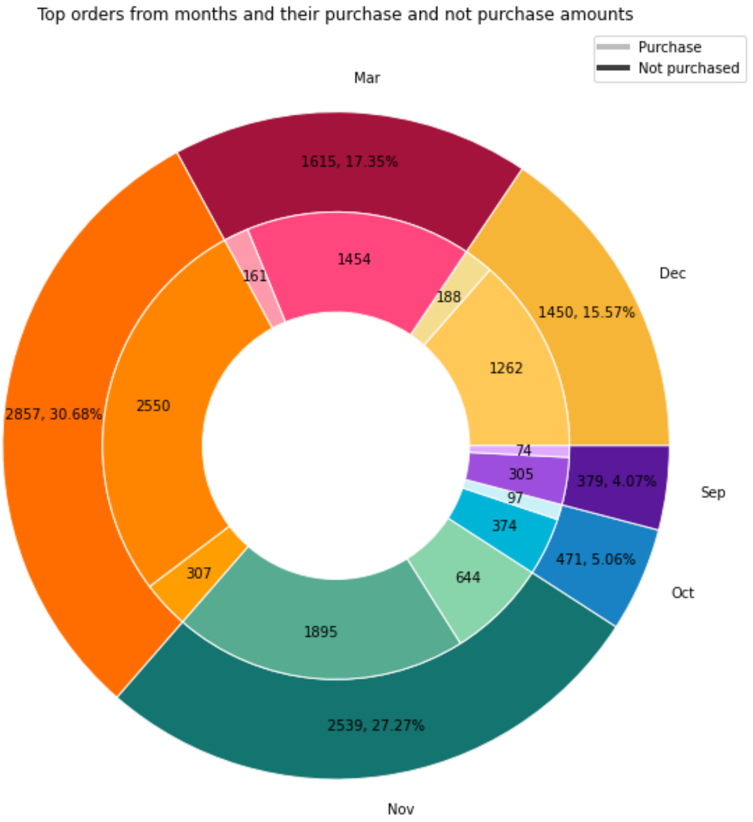
it into training and test sets can be quickly computed with the `train_test_s`

LABELS VIZUALIZATION

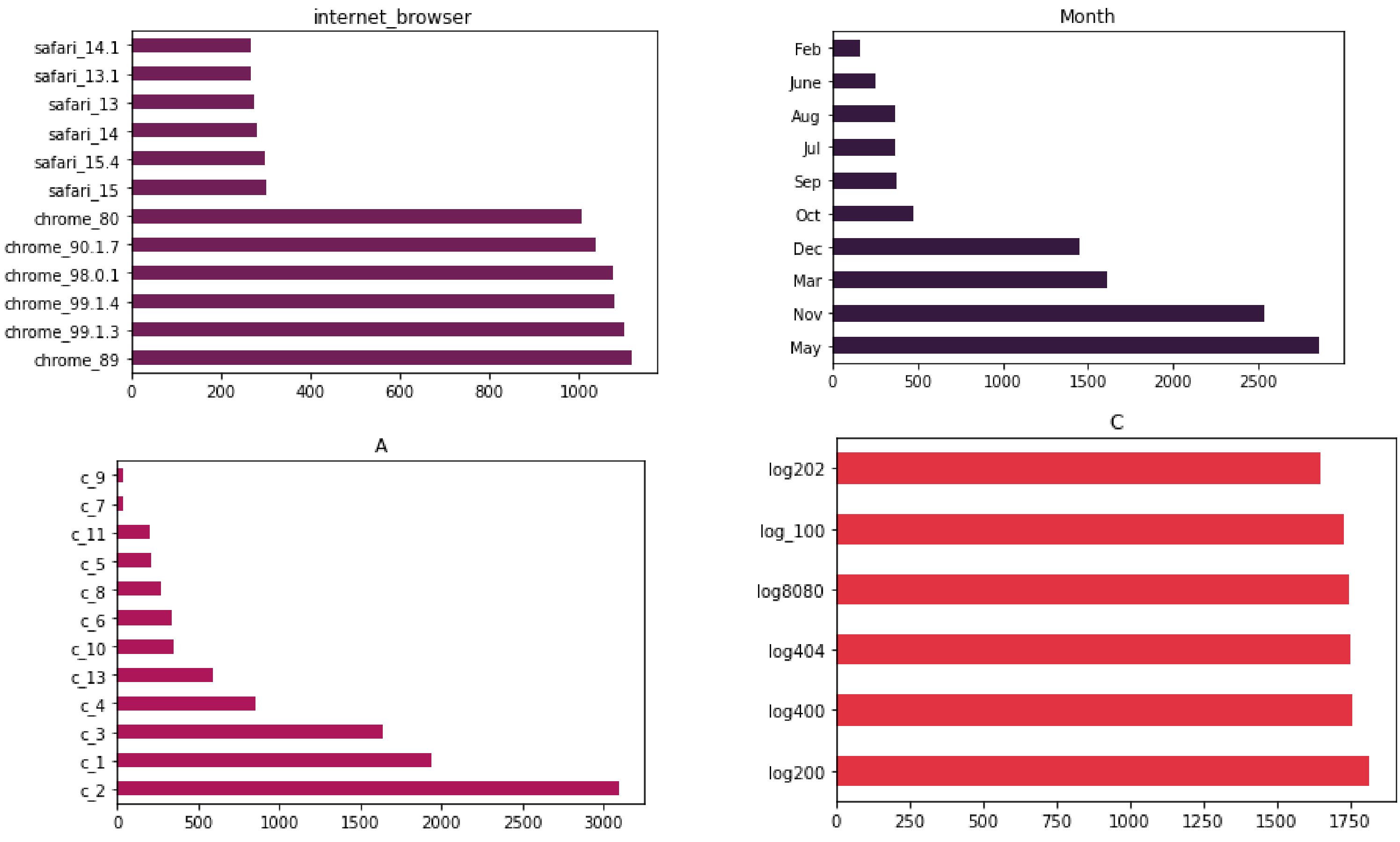


"MONTH" VIZUALIZATION

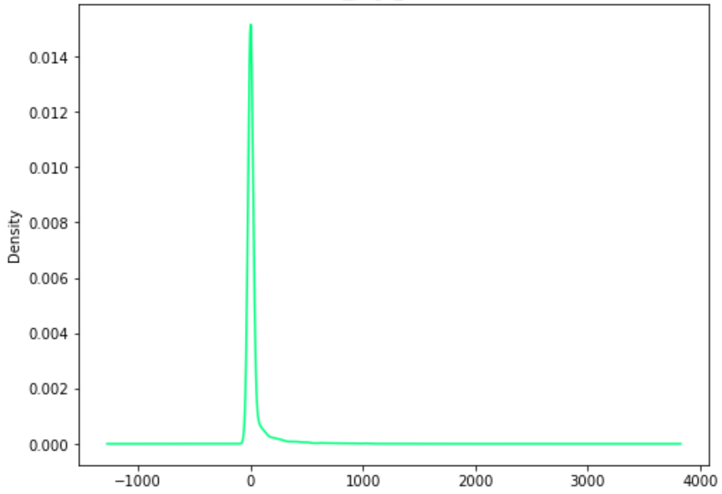




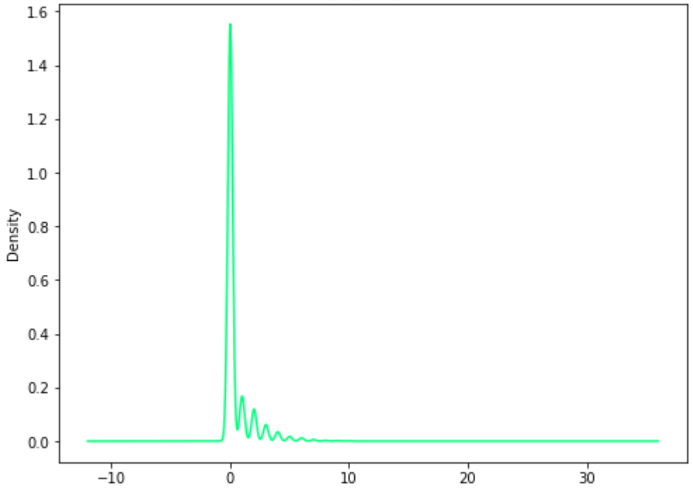
CATAGORIAL FEATURES DISTRIBUTION



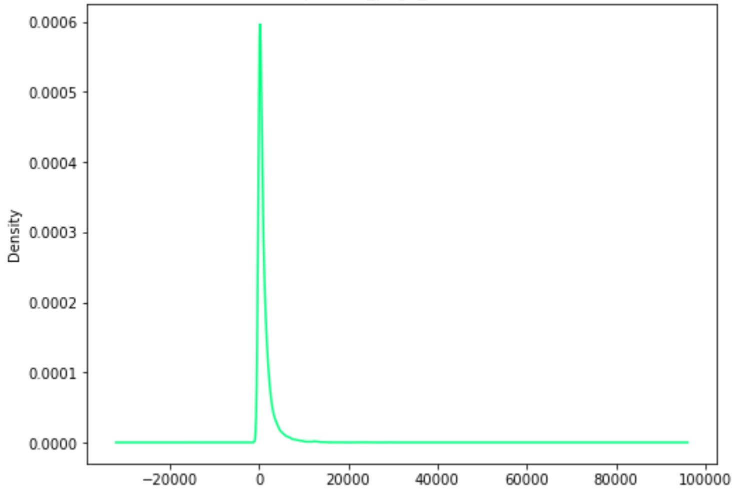
NUMERIC FEATURES DISTRIBUTION



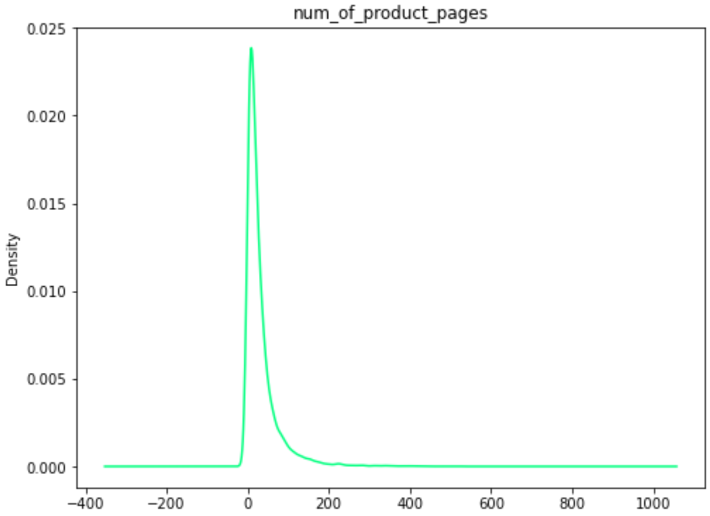
info_page_duration



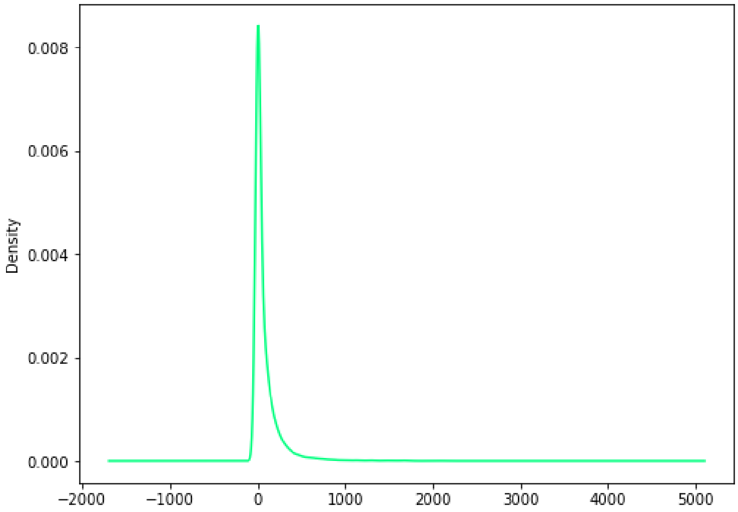
num_of_info_pages



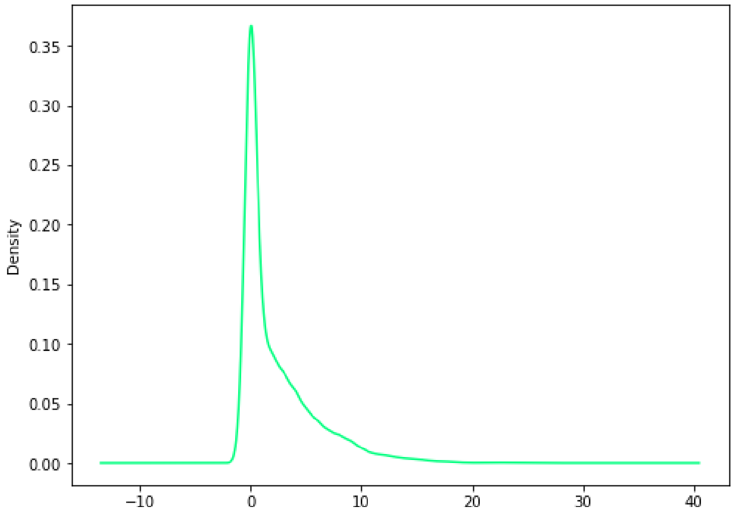
product_page_duration



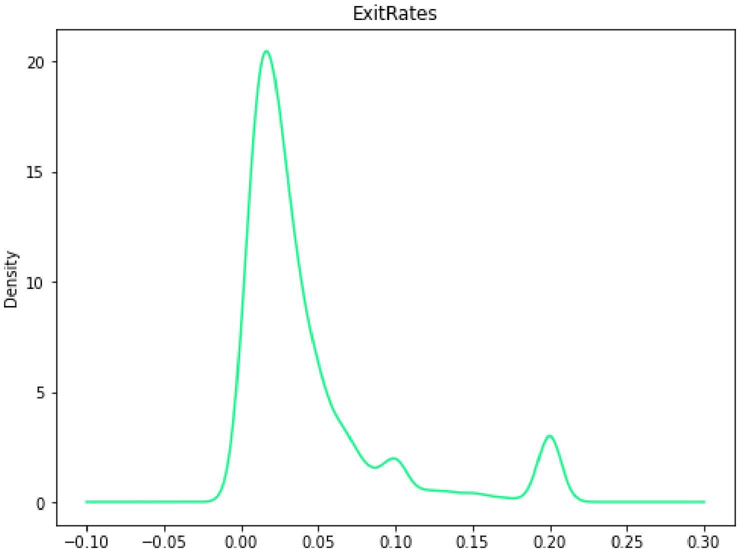
num_of_product_pages

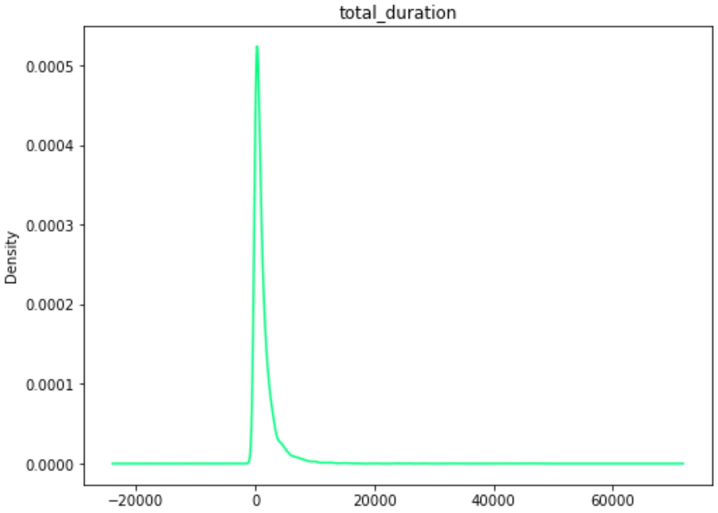


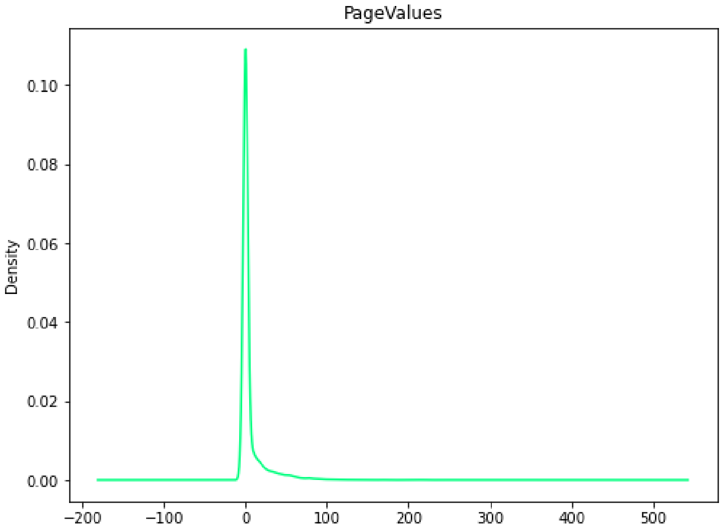
admin_page_duration

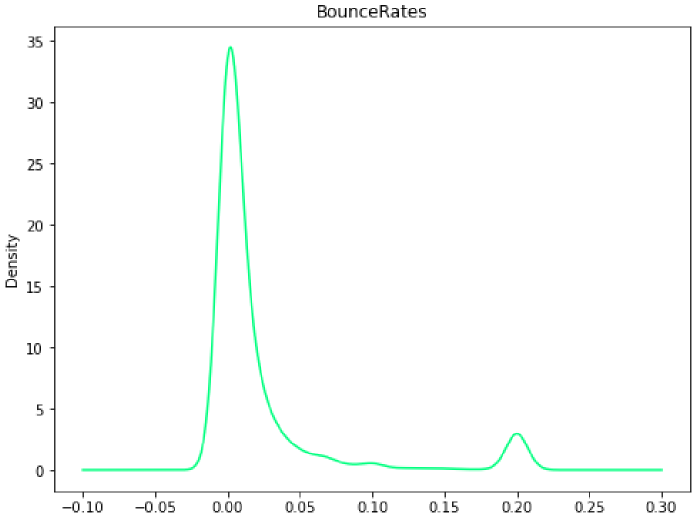


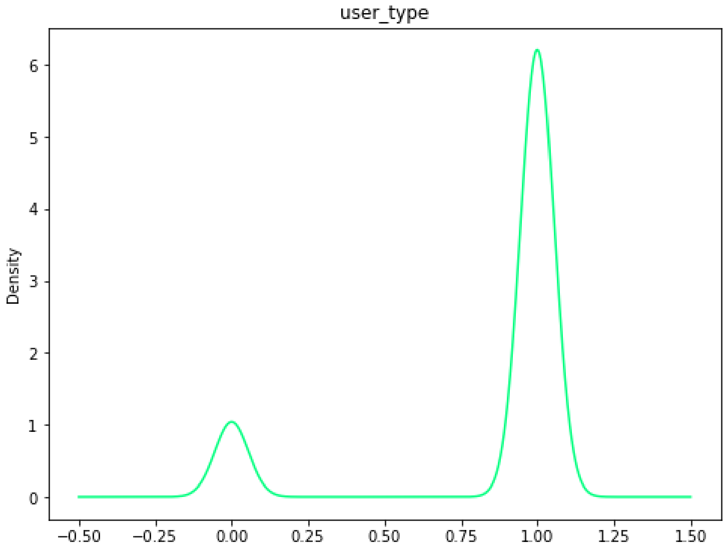
num_of_admin_pages

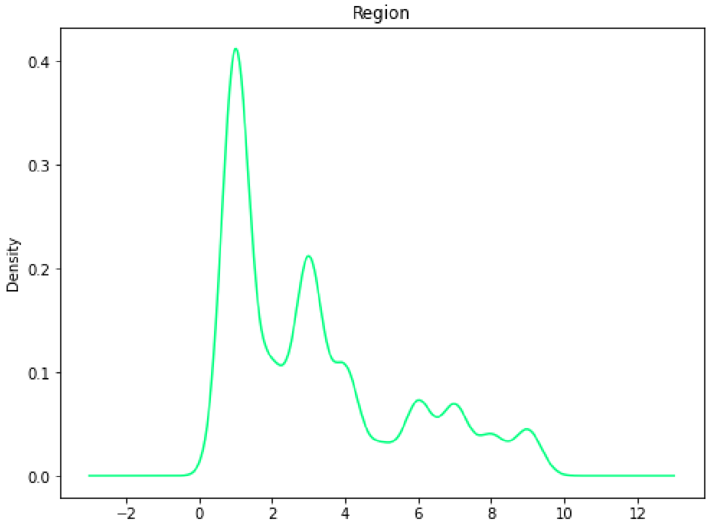

ExitRates
Density
0
5
10
15
20
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30

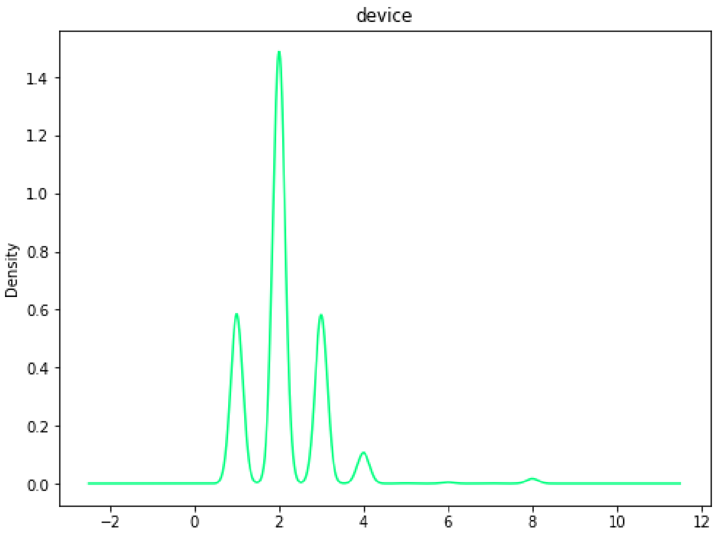

total_duration
Density
0.0000
0.0001
0.0002
0.0003
0.0004
0.0005
−20000
0
20000
40000
60000

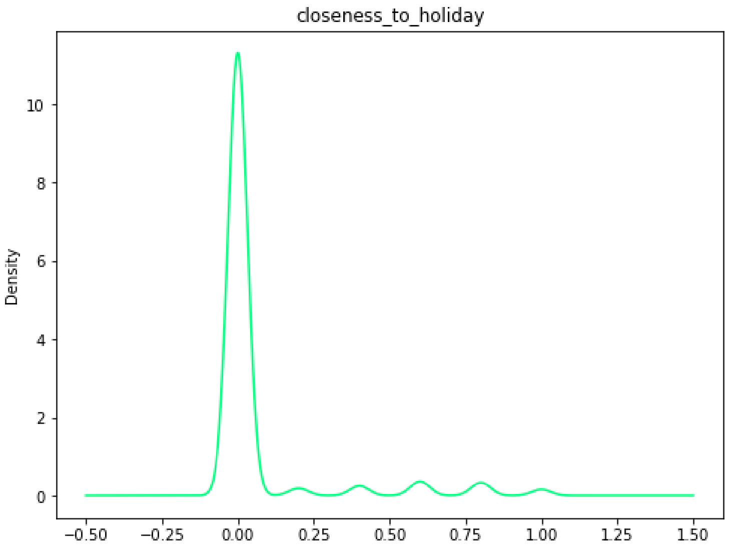

PageValues
Density
0.00
0.02
0.04
0.06
0.08
0.10
−200
−100
0
100
200
300
400
500

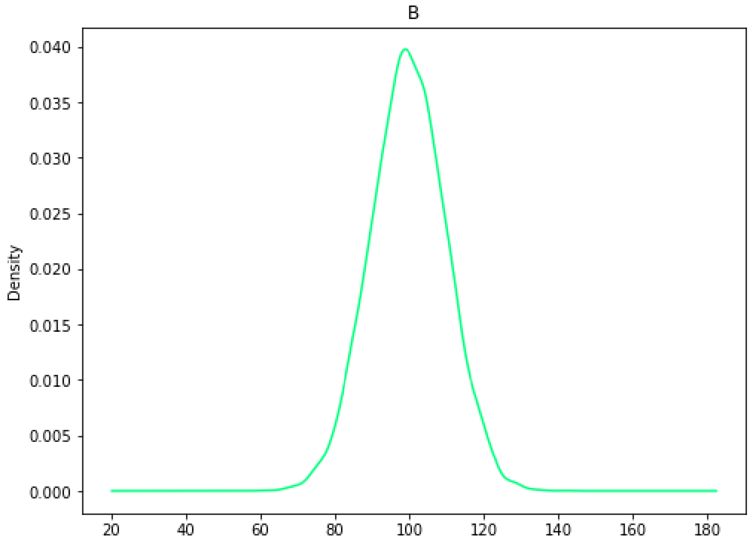

BounceRates
Density
0
5
10
15
20
25
30
35
−0.10
−0.05
0.00
0.05
0.10
0.15
0.20
0.25
0.30

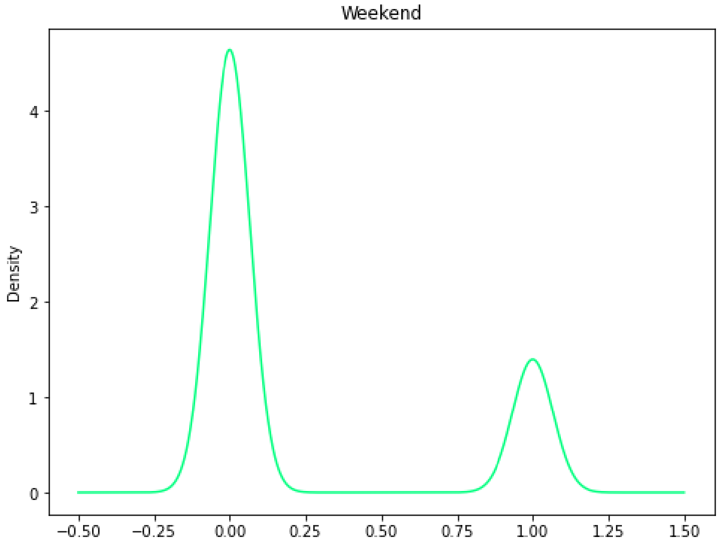

user_type
Density
0
1
2
3
4
5
6
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

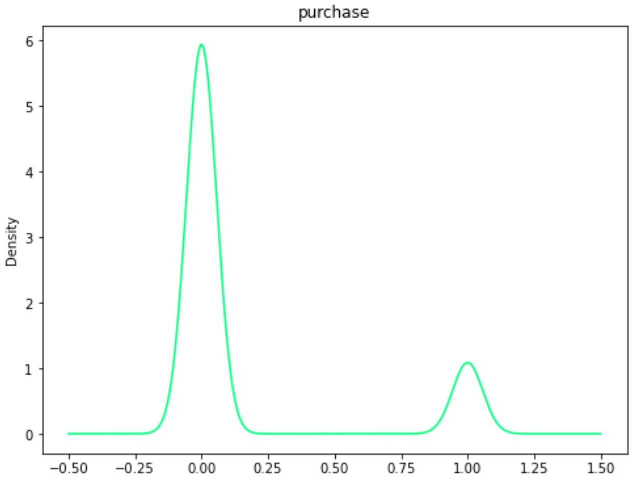

Region
Density
0.0
0.1
0.2
0.3
0.4
−2
0
2
4
6
8
10
12


device
Density
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
−2
0
2
4
6
8
10
12

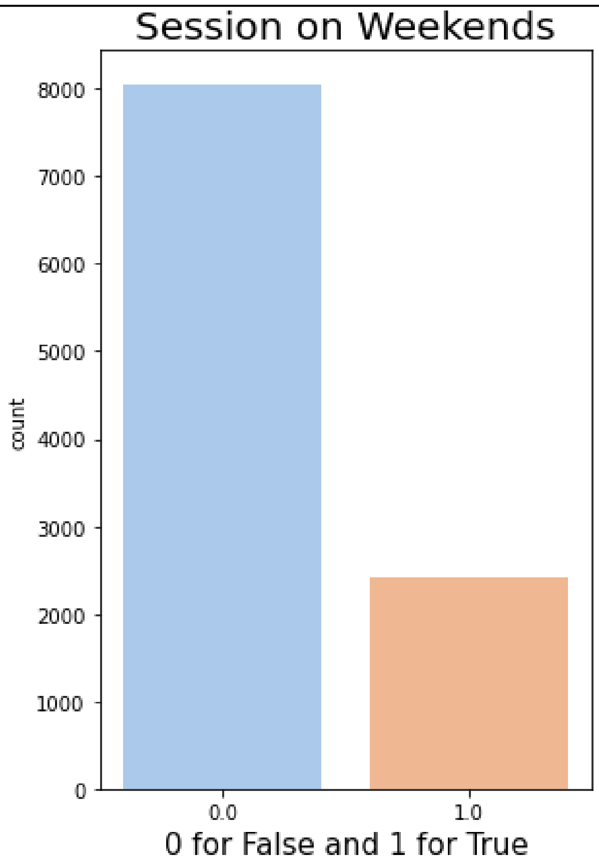

closeness_to_holiday
Density
0
2
4
6
8
10
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


B
Density
0.000
0.005
0.010
0.015
0.020
0.025
0.030
0.035
0.040
20
40
60
80
100
120
140
160
180

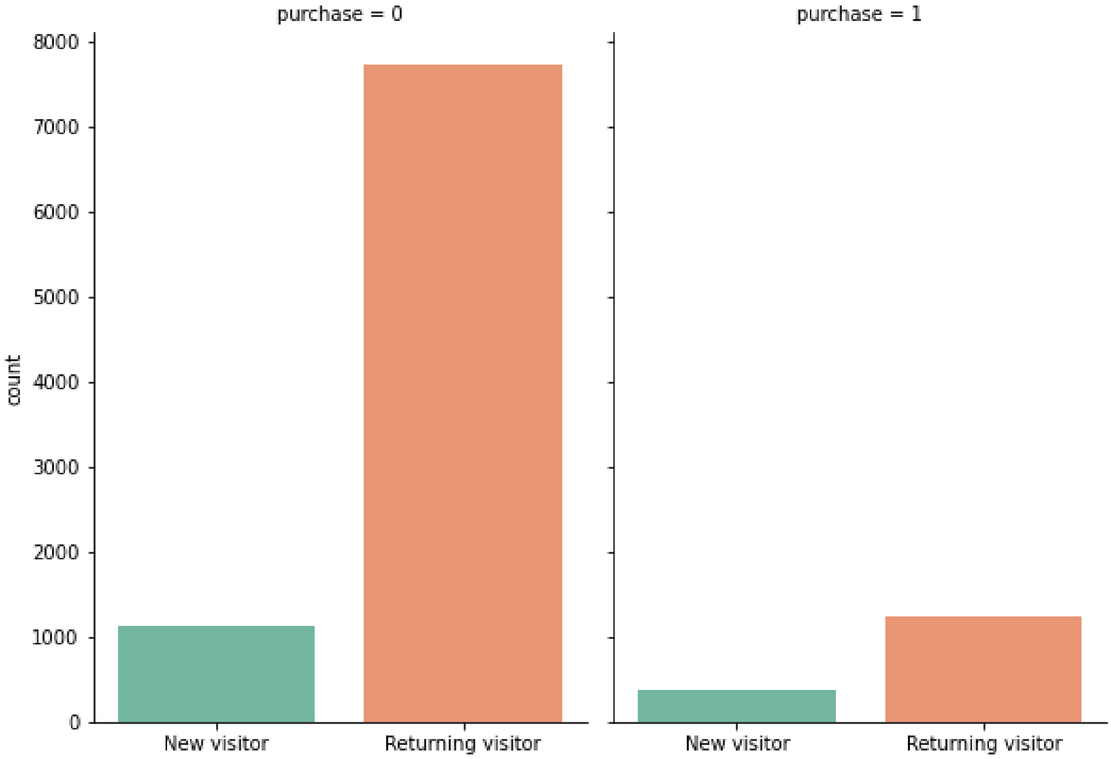

Weekend
Density
0
1
2
3
4
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50


purchase
Density
0
1
2
3
4
5
6
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
1.50

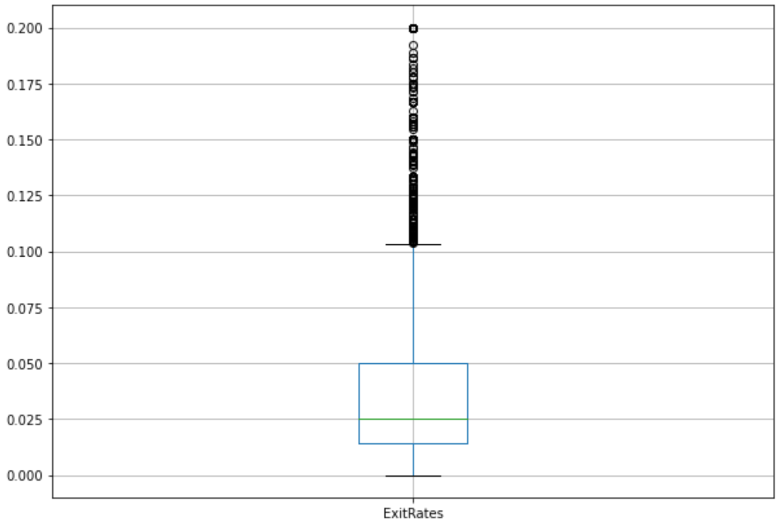
"WEEKEND" VIZUALIZATION

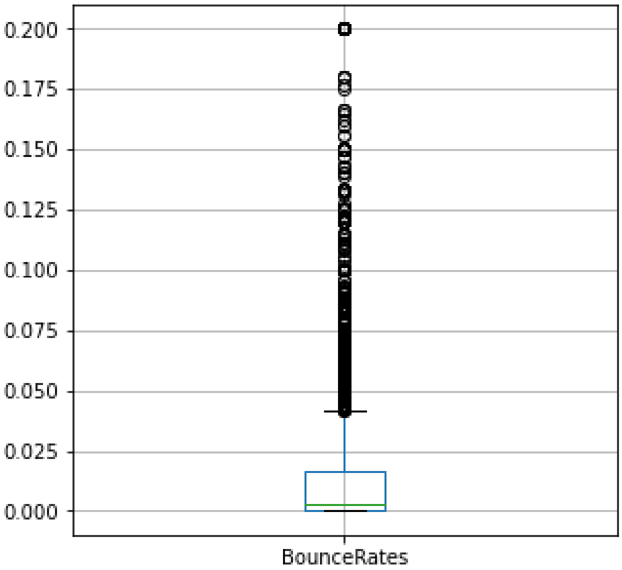


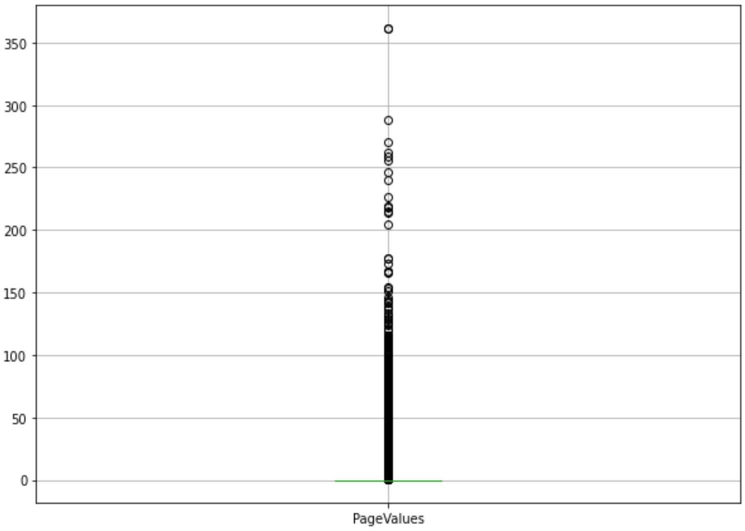
"USER TYPE" VIZUALIZATION

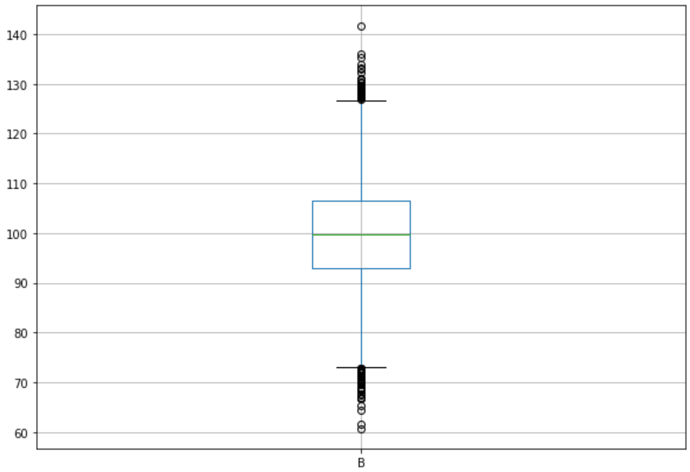


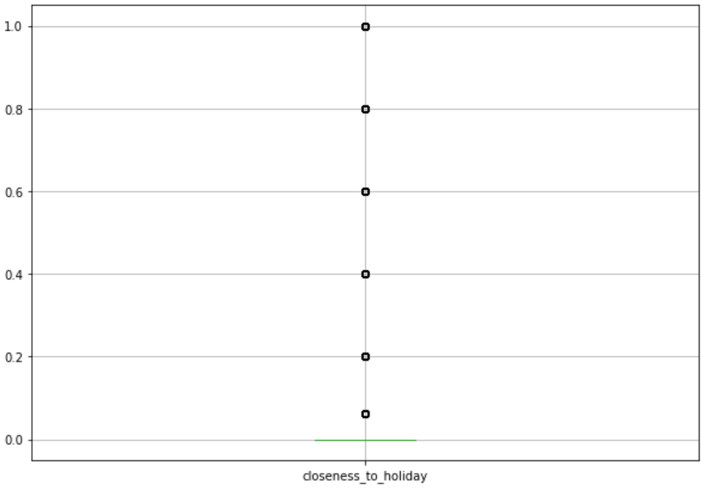
OUTLIERS VIZUALIZATION

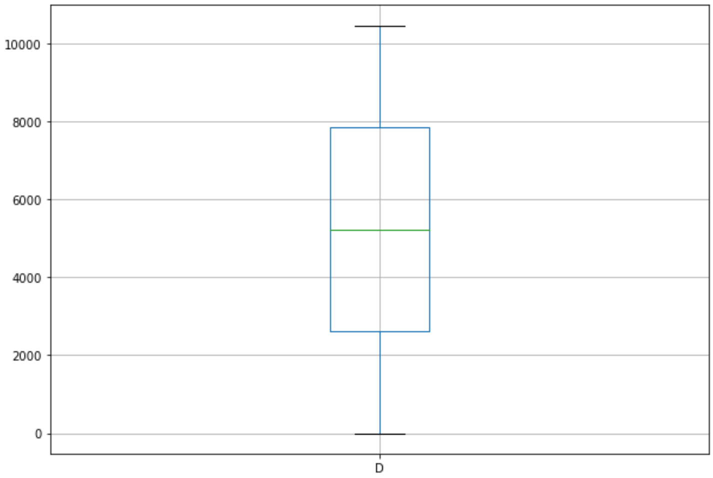


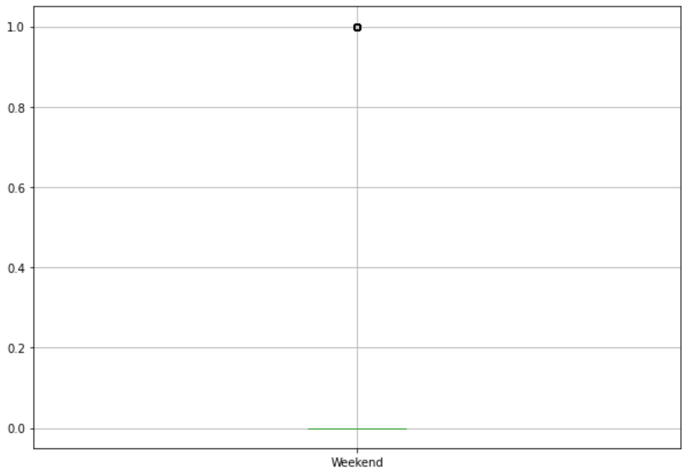


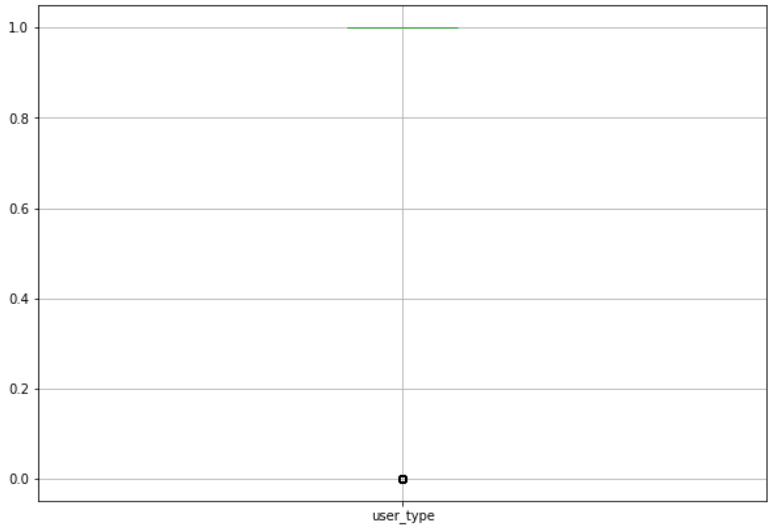


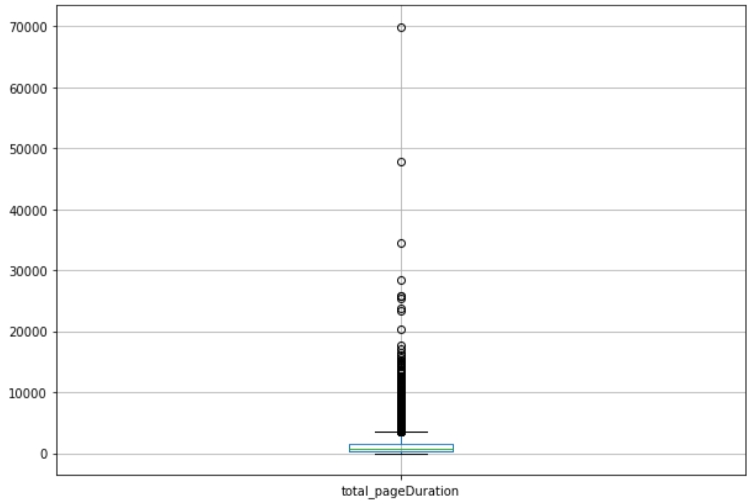


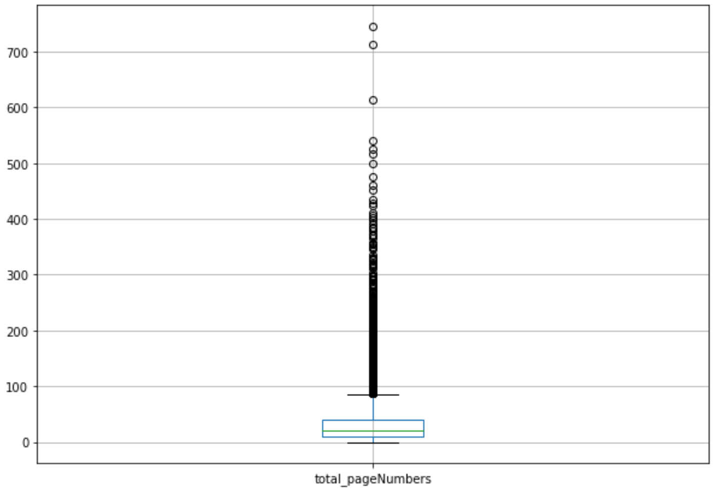


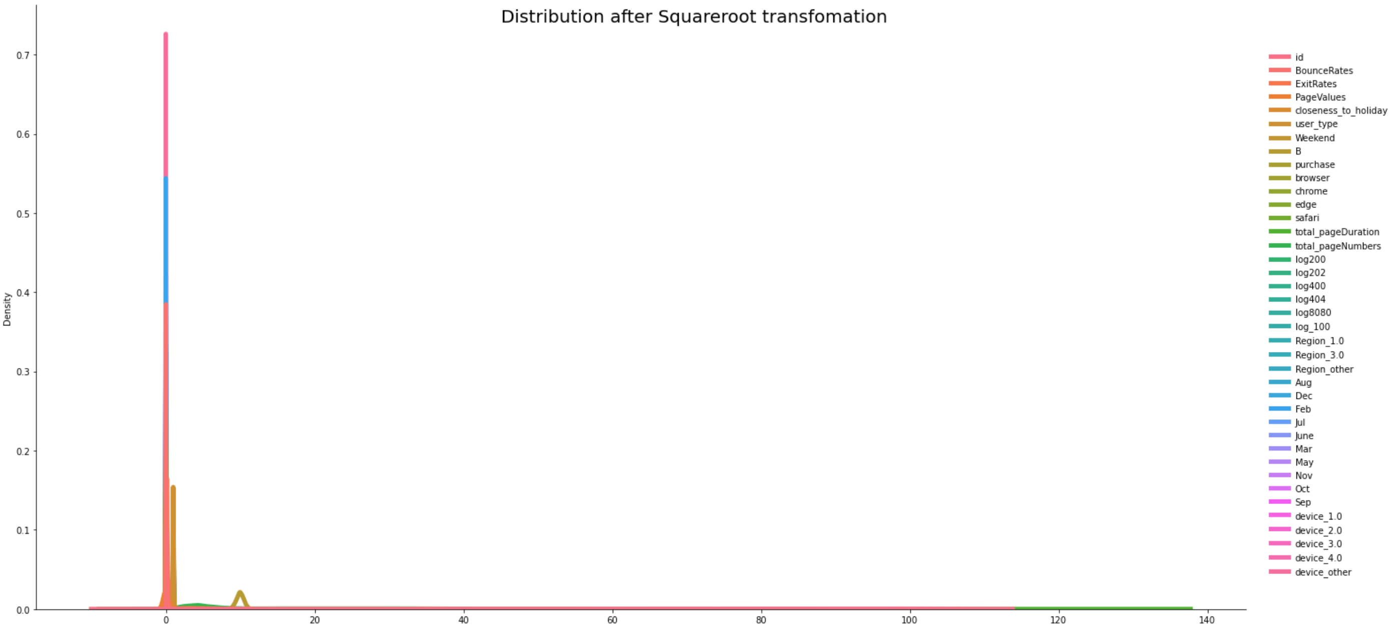


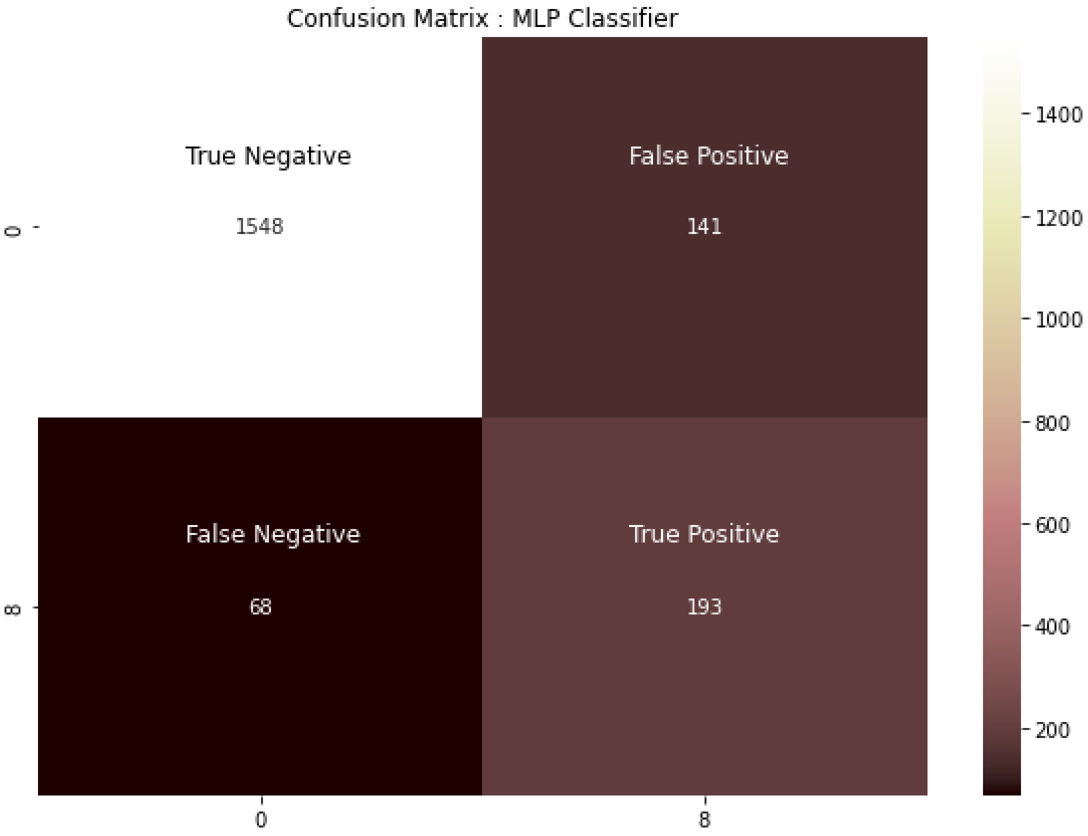

1.0
0.8
0.6
0.4
0.2
0.0
Weekend

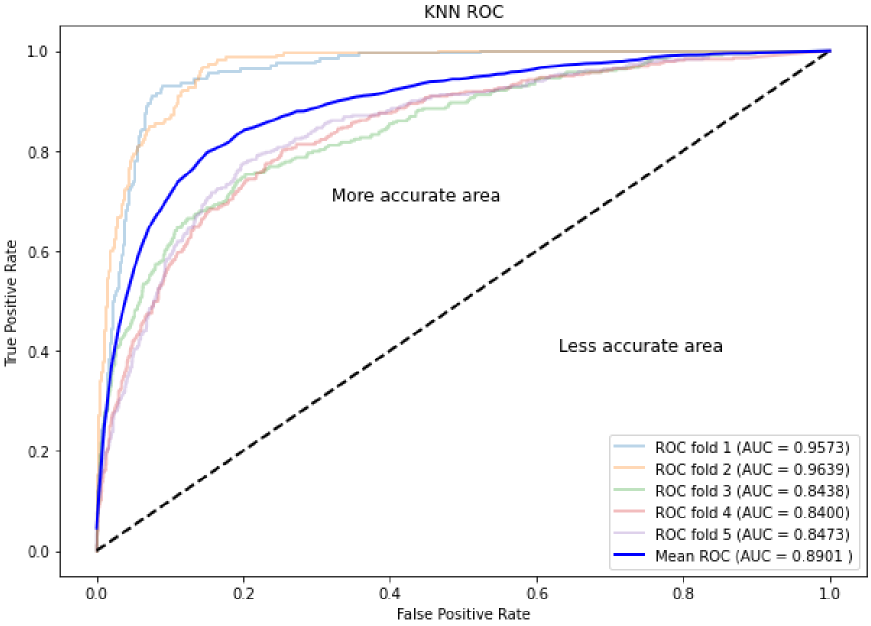

1.0
0.8
0.6
0.4
0.2
0.0
user_type

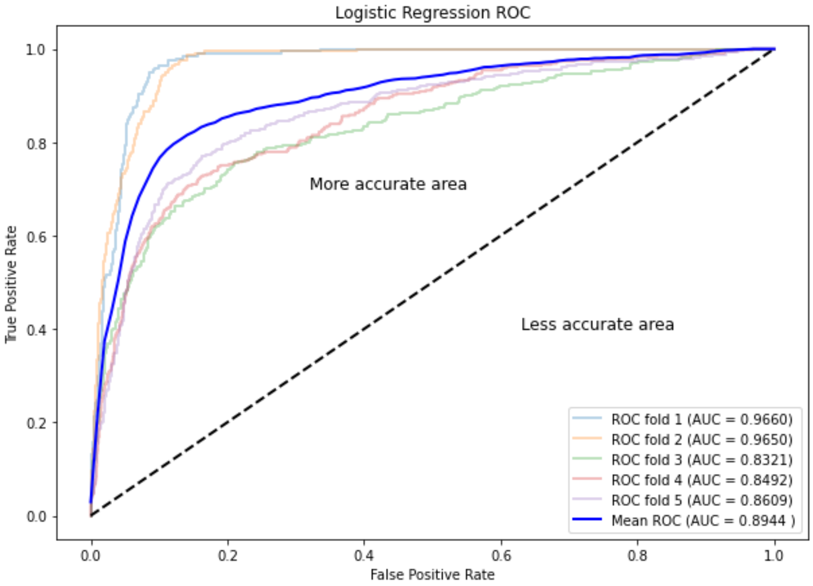

70000
60000
50000
40000
30000
20000
10000
0
total_pageDuration

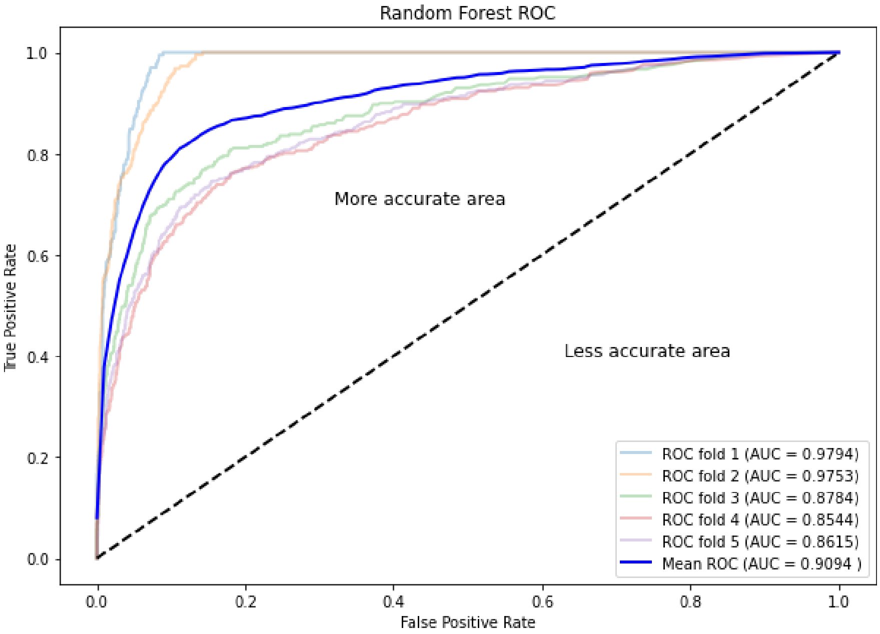

700
600
500
400
300
200
100
0
total_pageNumbers

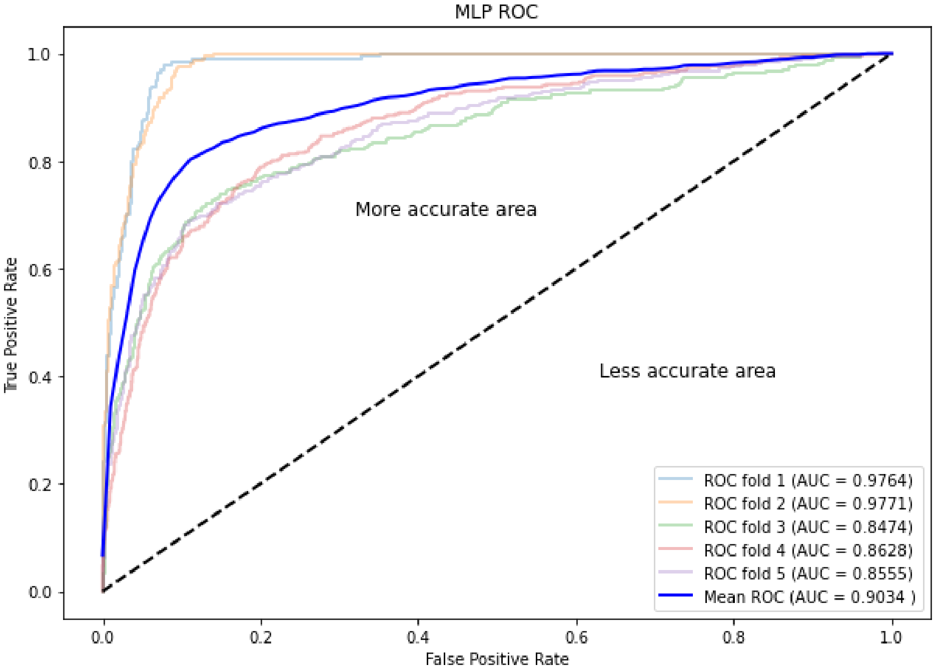

Distribution after Squareroot transfomation
Density
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
20
40
60
80
100
120
140
id
BounceRates
ExitRates
PageValues
closeness_to_holiday
user_type
Weekend
B
purchase
browser
chrome
edge
safari
total_pageDuration
total_pageNumbers
log200
log202
log400
log404
log8080
log_100
Region_1.0
Region_3.0
Region_other
Aug
Dec
Feb
Jul
June
Mar
May
Nov
Oct
Sep
device_1.0
device_2.0
device_3.0
device_4.0
device_other

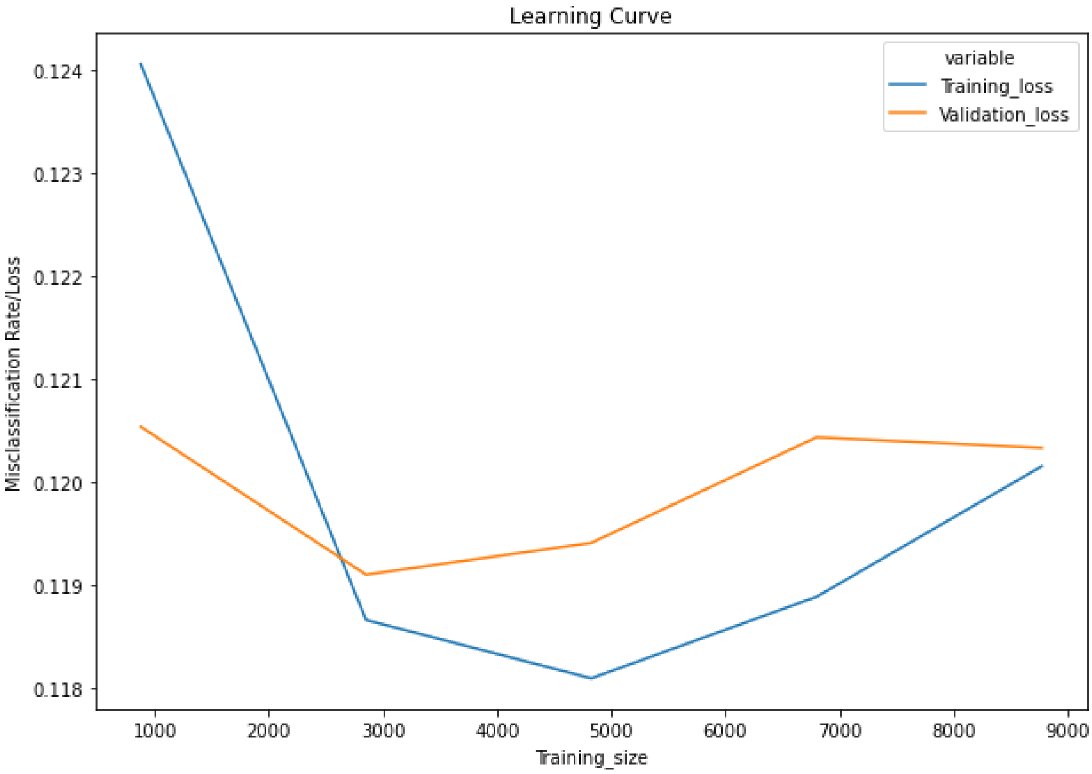

Confusion Matrix : MLP Classifier
True Negative
1548
False Positive
141
False Negative
68
True Positive
193
0
8
0
8
1400
1200
1000
800
600
400
200

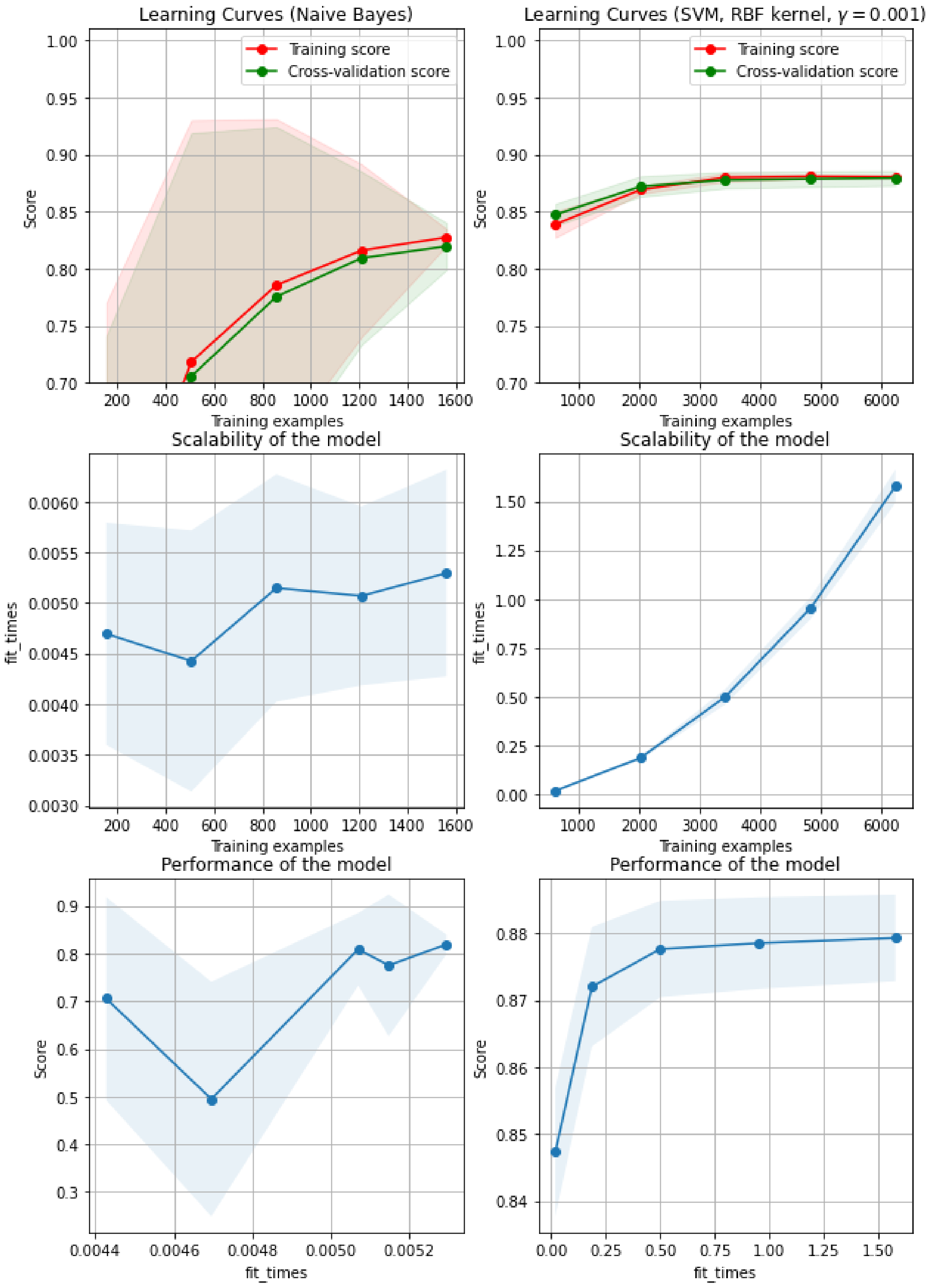

KNN ROC
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
More accurate area
Less accurate area
ROC fold 1 (AUC = 0.9573)
ROC fold 2 (AUC = 0.9639)
ROC fold 3 (AUC = 0.8438)
ROC fold 4 (AUC = 0.8400)
ROC fold 5 (AUC = 0.8473)
Mean ROC (AUC = 0.8901)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate


Logistic Regression ROC
1.0
0.8
0.6
0.4
0.2
0.0
True Positive Rate
More accurate area
Less accurate area
ROC fold 1 (AUC = 0.9660)
ROC fold 2 (AUC = 0.9650)
ROC fold 3 (AUC = 0.8321)
ROC fold 4 (AUC = 0.8492)
ROC fold 5 (AUC = 0.8609)
Mean ROC (AUC = 0.8944)
0.0
0.2
0.4
0.6
0.8
1.0
False Positive Rate


Random Forest ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
More accurate area
Less accurate area
ROC fold 1 (AUC = 0.9794)
ROC fold 2 (AUC = 0.9753)
ROC fold 3 (AUC = 0.8784)
ROC fold 4 (AUC = 0.8544)
ROC fold 5 (AUC = 0.8615)
Mean ROC (AUC = 0.9094)


MLP ROC
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
More accurate area
Less accurate area
ROC fold 1 (AUC = 0.9764)
ROC fold 2 (AUC = 0.9771)
ROC fold 3 (AUC = 0.8474)
ROC fold 4 (AUC = 0.8628)
ROC fold 5 (AUC = 0.8555)
Mean ROC (AUC = 0.9034)


Learning Curve
variable
Training_loss
Validation_loss
Misclassification Rate/Loss
0.124
0.123
0.122
0.121
0.120
0.119
0.118
1000
2000
3000
4000
5000
6000
7000
8000
9000
Training_size


Learning Curves (Naive Bayes)
Training score
Cross-validation score
Score
1.00
0.95
0.90
0.85
0.80
0.75
0.70
200
400
600
800
1000
1200
1400
1600
Training examples
Learning Curves (SVM, RBF kernel, $\gamma = 0.001$)
Training score
Cross-validation score
Score
1000
2000
3000
4000
5000
6000
Training examples
Scalability of the model
fit_times
0.0060
0.0055
0.0050
0.0045
0.0040
0.0035
0.0030
Training examples
Scalability of the model
fit_times
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Training examples
Performance of the model
Score
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.0044
0.0046
0.0048
0.0050
0.0052
fit_times
Performance of the model
Score
0.88
0.87
0.86
0.85
0.84
0.00
0.25
0.50
0.75
1.00
1.25
1.50
fit_times