Introduction to Machine Learning – Final Project

Team 34 – Michaela Segall (313562191), Avi Sananes (208057497)

Executive Summary

We were asked to predict when an Ecommerce session has ended with a purchase. After examining four Machine Learning models, the model that scored the best score is Random Forest, with AUC of 0.94. We used several techniques during the process, such as dimensionality reduction (with testing both PCA and Forward Selection), hyperparameters selection using GridSeaechCV, and K-Fold Cross Validation to reduce the chance of overfitting. The key knowledge we gathered during the exploration of the data helped us to standardize the data we have, fill its the empty values and detect and remove outliers.

Data Exploration

To really understand the data is probably the most important, difficult and exhausting part of a Data Science project. At first, we really wanted to have a better understating and to gain a domain knowledge of the data we are facing, and which features it contains.

In order to do so, we used several data exploration techniques. At first, we wanted to take a glimpse at the train data, so we used the head() function to see how the values in our data look like. At this point we already understood that some serious topics that should be addressed at the pre-processing stage, for example the fact that the duration columns have the word “minutes” in it, and that some of the values are missing.

We noticed that our train data has 10479 rows, which gives us plenty of room for outliers detection and removing, and for separating the dataset to train and validation sets in a later stage. In addition, we saw that we have 22 features (not including the “purchase” column), which we might want to reduce when testing our models.

Afterwards, we used both boxplot (figure 1) and histogram (figure 2) to see the data distribution. We noticed that due to high variation, some of the columns are not really fit the boxplot visualization, but it did help us to realize that some of the columns, like the region and total_duration columns, have obvious outliers. With the histogram visualization, we were able to see that most of the columns are either normally or log-normally distributed. This assumption will serve us for detecting and removing outliers.

Eventually, we used heatmap to plot the feature correlation (figure 3). We found out that some features, like BoundRates and ExitRates, are strongly correlated, and we might want to consider reducing features with this kind of strong correlation. In addition, we can see that some features are more correlated to “purchase” than other – like PageValues and the “D” column.

Once we get to the dimensional reduction part, we will take these conclusions into consideration.

Pre-processing

The first part of the pre-processing stage is to normalize the data and modify it in a way that would allow us to use it in machine learning models. At the exploration part of the project, we've reviewed the data manually and looked how the values look like.

As the modifications can have several methods of implementation (for example: categorical data can have different encoders), we've created a function that receives a dataframe and which columns should

go through which preprocessing. We've determined which processing is required through manually looking at the table and the distinct values of each feature. The preprocessing contains:

- Converting **boolean** data to 0,1.
- Converting **month** names to numbers. (which we eventually decide not to use)
- Extract **numbers** from string with words = "23.5 minutes".
- **Browser data** - looking carefully on the values, there is a pattern detected - "<browser_name>_<version>" where the version can be in different patterns and styles. We extract the browser name and later encode it, and the browser version we take only the "major" and ignore the "minor" version part.
- **Categorical data** – we decided to convert the categorial columns to Dummy variables. This will add us several new features to consider, but we will try to deal with it with dimensionality reduction.

As we finished setting our data with numerical values, we were able to move on to dealing with missing values. During the data exploration process, we noticed that all the columns in our train data have missing values (NaNs). We decided to use different approaches for each column:

- **Column D** – we decided to remove this column completely, after realizing that most of it is empty and it doesn't correlate with the "purchase" column.
- **Page durations** – first, we wanted to make sure to avoid conflicts between columns. We filled the page duration columns with zeros wherever the total duration equals zero, wherever the number of visited page equals zero and vice versa.
- **Column A** – we noticed that the column values are between 1-15, so we filled the missing values with zeros.
- **Column C** – since it has very little amount of missing data, we filled it with the most common value.
- At this point, we used **KNN** to fill any remaining missing values in all the other columns, except the total duration.
- At last, we filled the **total duration** with the sum of the other page duration fields.

The last thing we've done in the pre-processing stage is to remove outliers. Zscore test usually applied on normally distributed data. As we previously seen, not all our columns are normally distributed, but we believe that this will be covered due to the large amount of train data that we have. In addition, we will look at the "before and after" charts, and make sure that only obvious outliers are filtered out for each column. We used Zscore with 5.5 Standard Deviations, so we won't remove more than 10% of the training data we have.

Now that we have our data standardized and filled, we wanted to test it on a model. We ran a Logistic Regression model for the first time on our data, and the results were rather surprising.

The Logistic Regression model got a ±0.89 AUC score on our validation data, which was 20% of our training data. We were concered of overfitting but the fact the we got the same good result on both the validation and training set indicate the model isn't suffering from overfitting.

With that said, we are still going to take some preventative steps to reduce the change of overfitting:

1. **Dimensionality reduction** - reducing the dimensionality of the data reduces the complexity of it, as more complex models may over fit the data (we had X features after the normalization).
2. **Regularization** - we will use regularization methods such as Lasso or Ridge to reduce the model's complexity. Regularization adds penalty to the model as the complexity increases.

3. **K-Fold cross validation** - althogh cross validation doesn't prevent the model from being over fitted, it could help us to see if it is suffering from overfitting. Combining it with the other methods will reduce the concern of our model being over-fitted.

Dimensional reduction

One of the measures that we can take to reduce overfitting is dimensional reduction. We previously noticed that we got many features, with some correlations between them. This led us to try and use to approaches - "logical" reduction and a "computed" one:

- **Logical reduction:**
  - High correlation - features that are very similar can be redundant and we may prefer taking only one of them instead of two. We saw that BounceRates and ExitRates are strongly correlated and decided to remove the BounceRates feature.
  - Logical dependency - all the duration columns are summed into a "total duration" column. Therefore, we can either use only the total, or its three separate components and then ignore the "total" column as redundant. We decided to use only the total duration column.
- **Computational methods** - we would test 2 different methods for dimension reduction and measure their impact on predicting with linear regression as a test case. The two models we'll tests:
  - Forward selection - compare each number of features to the one before it and pick the best number of features subset.
  - PCA - variance of 0.99 was chosen as a const for not losing too much of the data. Different consts would yield different results and it is configurable in change this assumption changes.

We will run the test on each model we're about to test since we realized that each model has different assumptions on the data, and the same method isn't best for every model. We will use the best selected approached for each model.

Model implementations and estimations

Now that we dealt with all the necessary perquisites, we can now run some models on our data and test their predictions. We chose to run the tests on Logistic Regression, KNN, Random Forest and MLP:

- **Logistic Regression:**
  - First, we saw that the best dimensionality reduction method for Logistic Regression is PCA. We fitted the PCA model on our data to match this selection.
  - We looked for the best "C" value using the "score" metric. It will help us avoid overfitting. The "C" value is the inverse of the regularization strength - as the value is smaller, the stronger the penalization. We'd like to choose the smallest "C" value that gives the best score. The result we got is 0.063. For the rest of the params we used default values as we thought C's impact is the largest.
  - We ran the model with the best "C" value and "l2" penalty. We used 8-fold cross validation to calculate the result.
  - We received a mean AUC score of 0.9, which is slightly better from the first time we run it. We got the same AUC score on the train data, which reduces the chance of overfitting.
- **KNN:**

- This time, the dimensionality reduction method that was selected is Forward Selection which came with the best 21 features.
- To find the best K value to use, we plotted for each K its MSE result. The K value that yielded the best MSE result is 7. We wanted to choose the lowest possible value to reduce the complexity.
- We ran the KNN model with 7 neighbors, and got an AUC score of 0.93, which is better than the Logistic Regression model.

- **Random Forest:**
  - Once again, the method selected is Forward Selection, with these columns: *['num_of_admin_pages', 'PageValues', 'closeness_to_holiday', 'Weekend', 'device_1.0', 'device_4.0', 'device_5.0', 'device_6.0', 'device_7.0', 'device_8.0', 'device_other', 'user_type_Other', 'user_type_other', 'browser_name_unknown', 'Month_Feb', 'Month_June', 'Month_Mar', 'Month_May', 'Month_Nov', 'Month_Sep', 'Month_other']*
  - This time, we used GridSearchCV to help us find the best hyperparameters for the model. These are the hyperparameters we looked for, as we find them most central for model improvement:
    - **n_estimators** - number of trees in the forest.
    - **max_depth** - depth levels of each tree.
    - **min_samples_split** - minumum samples required for a split in the tree.

    The result we got is *{'max_depth': 9, 'min_samples_split': 3, 'n_estimators': 90}*
  - The mean AUC score that we got for Random Forest is 0.94, which is the best score so far. We believe that the reasons for it are the fact that this is a more complex model, and that we used GridSearchCV to find the best hyperparameters.
  - Feature importance – Random Forest allows us to see which are the best features that helped for the model estimation. We added a plot (figure 4) and found out that these features had helped the most:
    - **PageValues** - according to the model, this is the most important feature. This feature indicates an average value for a page that a user visited, before landing on the goal page or completing a transaction. Since we are trying to predict which e-commerce sessions are ending with a purchase, it makes a perfect sense that higher PageValues will be given to sessions that eventually ended up with a purchase. In addition, according to the Google Analytics PageValue explanation, a PageValue of 0 indicates that the session didn't end with a transaction - which also makes it an extremely strong feature to predict purchases.
    - **num_of_admin_pages** - although it's less obvious than the PageValues feature, it is understandable that users, especially returning visitors, that are taking care of their app/website settings and profile are more engaged users, with a higher chance of purchasing items at the end of their sessions.
    - **Month_nov** - November is the month that people prepare to the holidays season. With Thanksgiving happening on November 24th, and Christmas only a month later, November is the last month to prepare to these events. According to this infographic by Yesmail (https://www.rcs-uk.com/people-start-shopping-for-christmas/#shopping-infographic), 31% of the people start their shopping for Christmas in November.

    - **closeness_to_holiday** - basically speaks for itself. People tend to but more when the holidays are approaching. According to this article by Finical (https://finicalholdings.com/us-holiday-shopping-statistics/), in 2020 holiday shopping made up 19.5% of total annual retail income.
- **Multi-Layer Perceptron:**
    - This time, due to a long running time, we didn't compare the dimensionality reduction methods, and instead ran the model twice – once with PCA and once with the same Forward Selection result of Random Forest.
    - We again used GridSeaechCV to find the best hidden_layer_sizes (number of neurons in the hidden layer) and alpha (the strength of the Ridge penalization).
    - With PCA, the GridSearchCV results were *{'alpha': 1, 'hidden_layer_sizes': (80,)}*, and it yielded a 0.93 mean AUC.
    - Forward Selection on the other hand, selected *{'alpha': 1, 'hidden_layer_sizes': (40,)},* and scored an AUC of 0.92.

Predictions

After running all the models, it appears that the model that scored the best AUC score is Random Forest, with AUC of 0.94 (figure 5). We will use it as the final model, using only the features from Forward Selection results and with the hyperparameters described above.
At this point, we implemented the final pipeline that extracts the CSV file with our model's predictions. The pipeline calls all the functions that we've built along the way to run it all at once.

Summary

- **To explore the data**, we used some basic DataFrame functions to see its values, and histogram and box plots to see the columns distribution.
- We used a **correlation matrix** that helped us find features that are strongly correlated, and we can reduce in later stage.
- We **normalized the data** to make it numeric and removed unnecessary characters.
- Afterwards, we used several methods **to fill in all missing values**, including KNN for most of the fields.
- We assumed that most of data is normally distributed and used z-score test to **remove outliers**.
- We decided to test two **dimensionality reduction** methods on each model, and for each to use the one that yields the best results.
- We ran Logistics Regression, KNN, Random Forest and MLP. Random Forest has yielded the best AUC score, with value of 0.94.
- We than decided to use Random Forest as our model for the final pipeline.

Appendix

Division of responsibility

- Data exploration - Together
- Data standardization - Michaela
- Data completion – Michaela
- Removing outliers - Avi
- Dimension reduction – Michaela
- Evaluating basic models and hyperparameters – Avi
- Evaluating complex models and hyperparameters – Avi
- Estimation utility – Michaela
- Final pipeline code - Michaela

Link to our github project: https://github.com/mikisegall/ml_final_project

Interesting plots

Figure 1 – Boxplot:

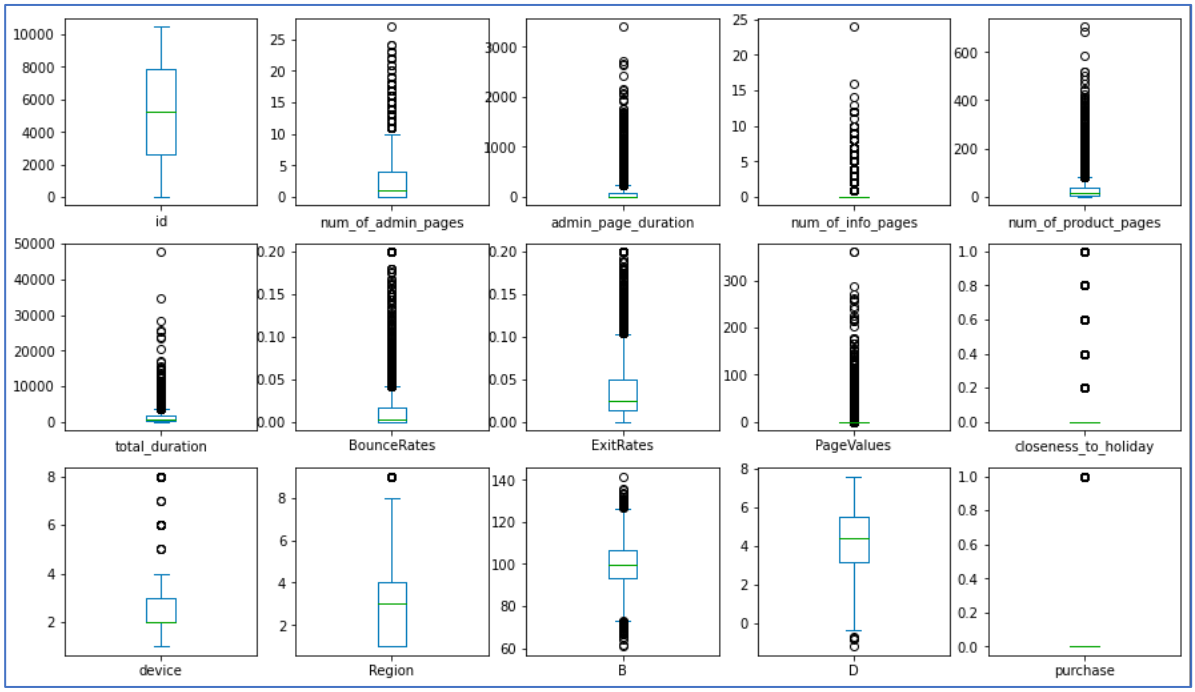

Figure 2 – Histogram:

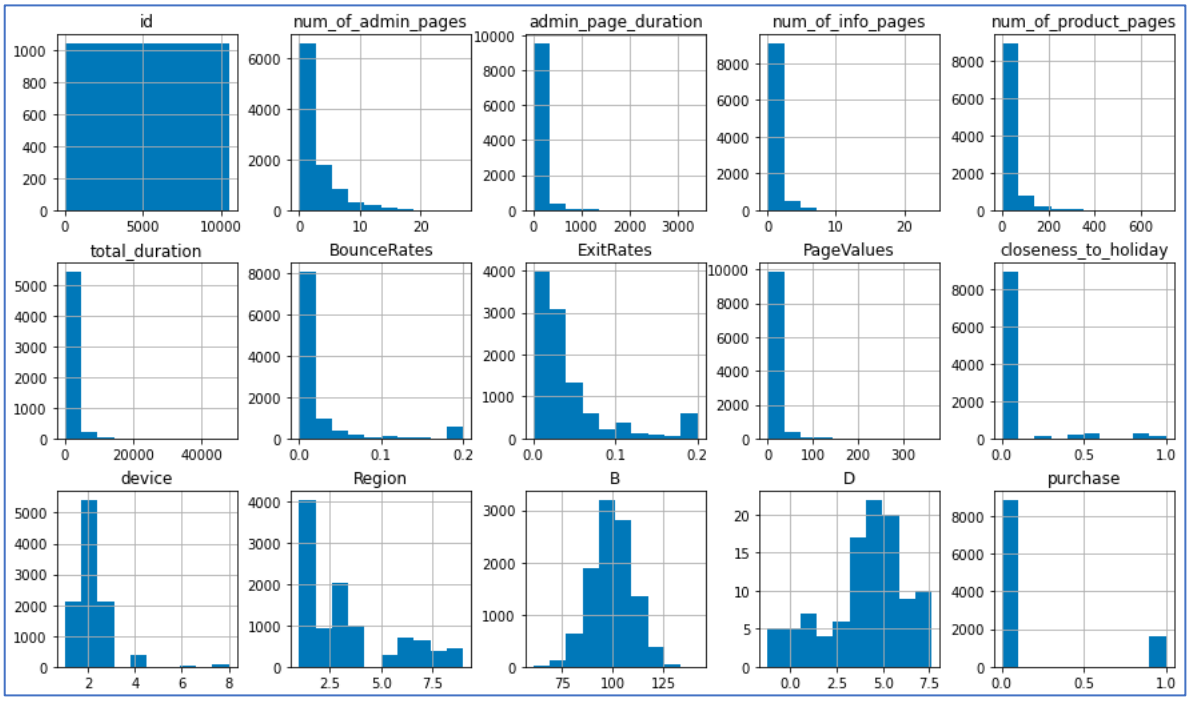

Figure 3 – Correlation plot:

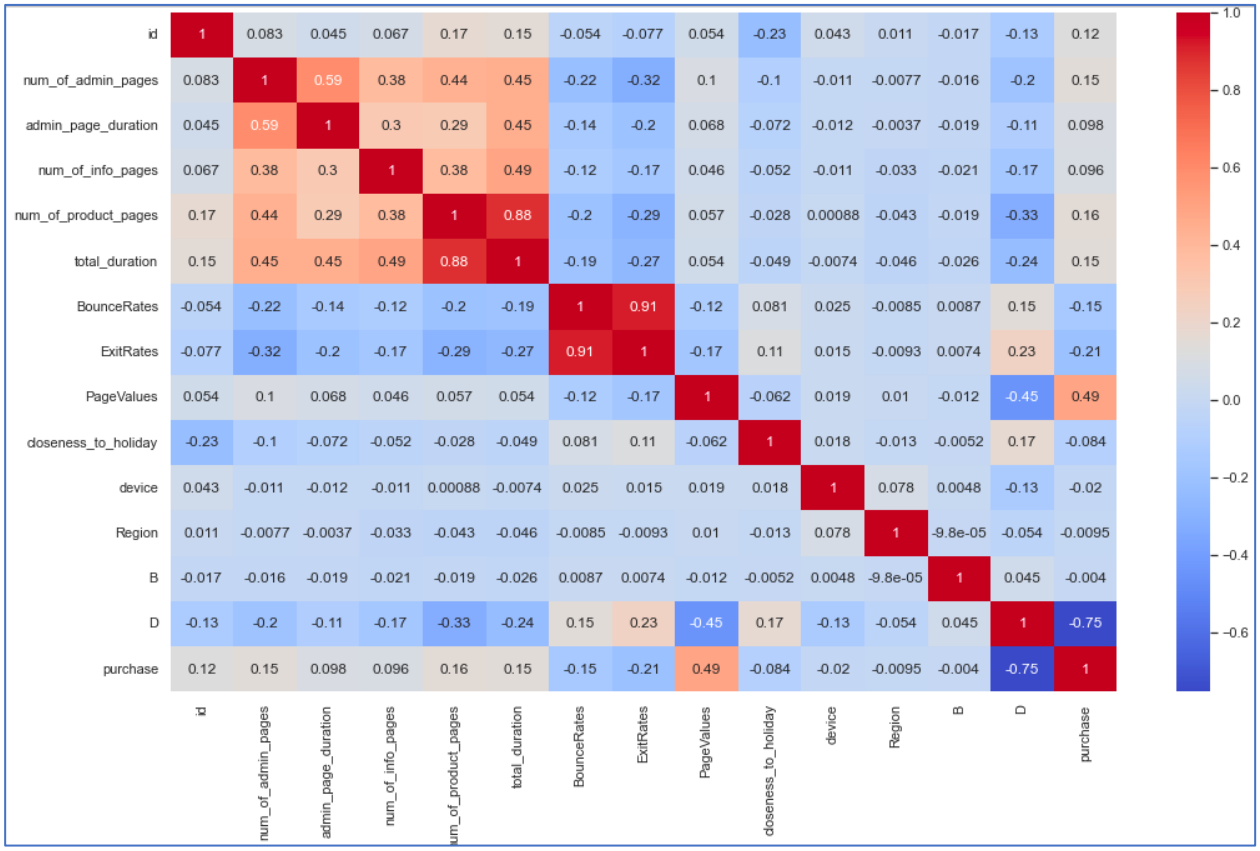

Figure 4 - Feature importance from Random Forest model:

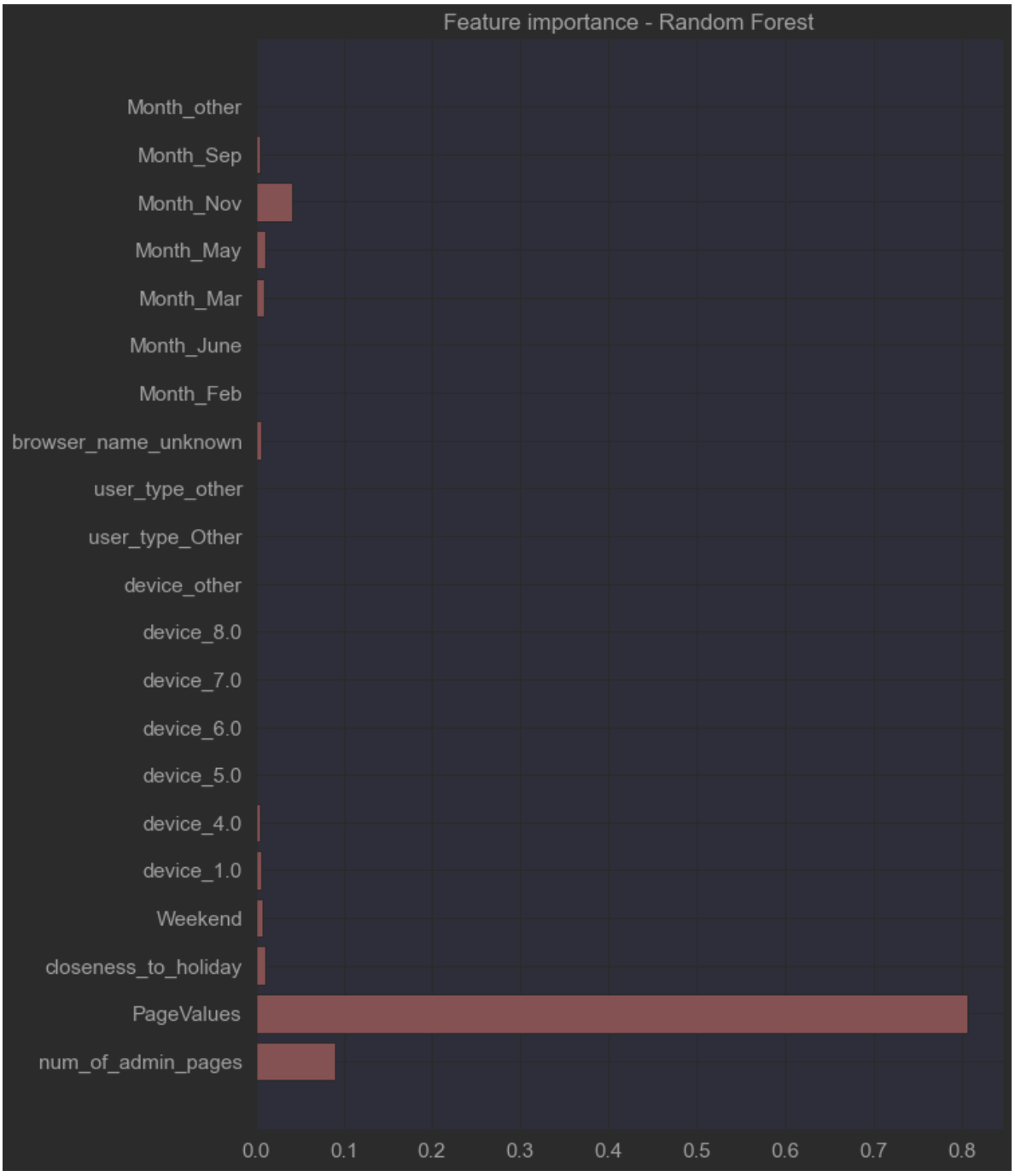

Figure 5 – Chosen model ROC Curve: Random Forest: