



Online Shoppers Purchasing Intention Analysis

Kevin LIM

Dataset informations and Features

The dataset consists of 10 numerical and 8 categorical features.

The 'Revenue' attribute is used as class label.

The dataset is clean, there are no missing values but the dataset is unbalanced. There is a risk of bias, so the analysis have to take the unbalanced dataset into consideration.

Features correlation matrix

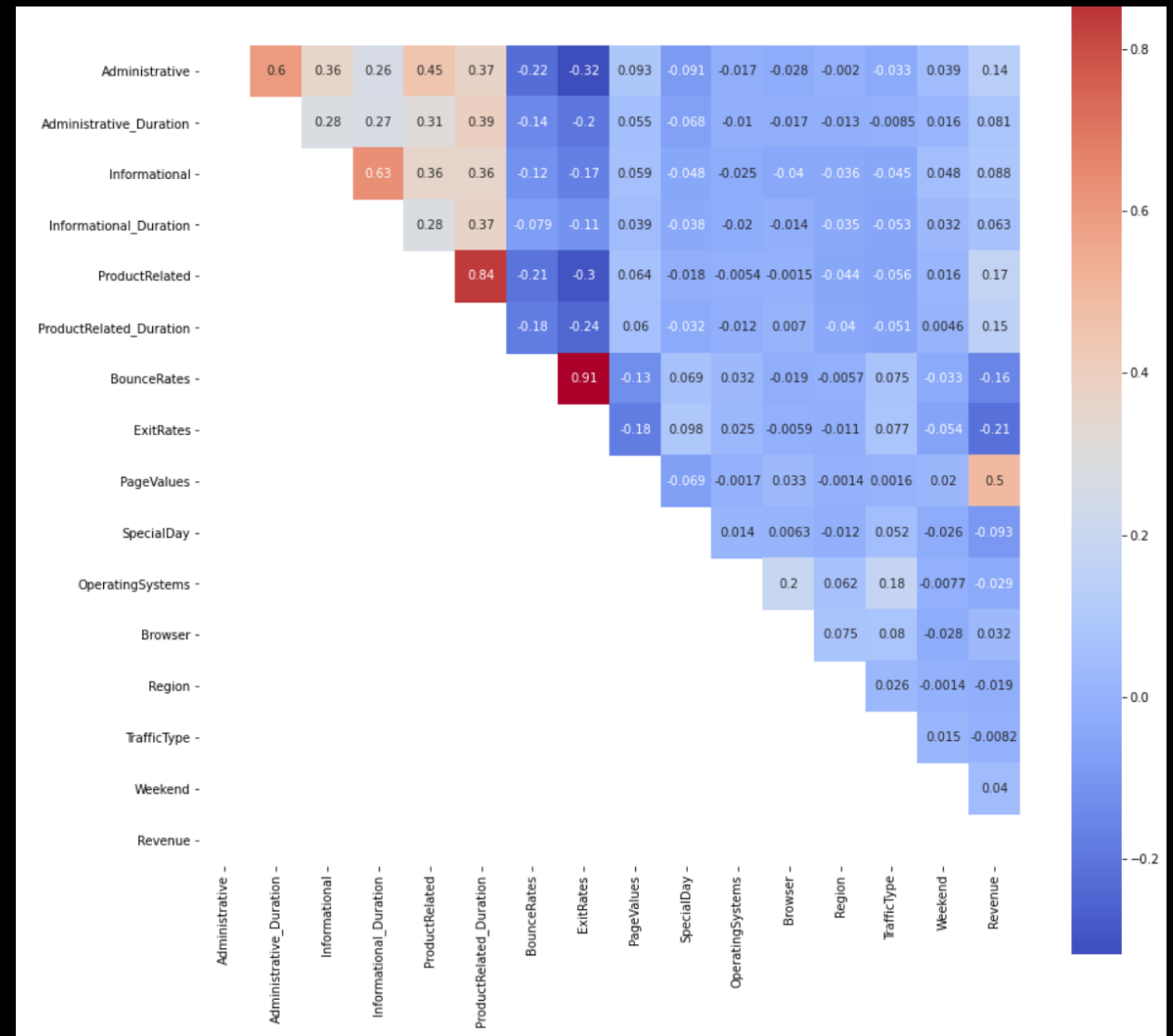
We see in the last column that the target is only correlated to a small number of variables in this dataset.

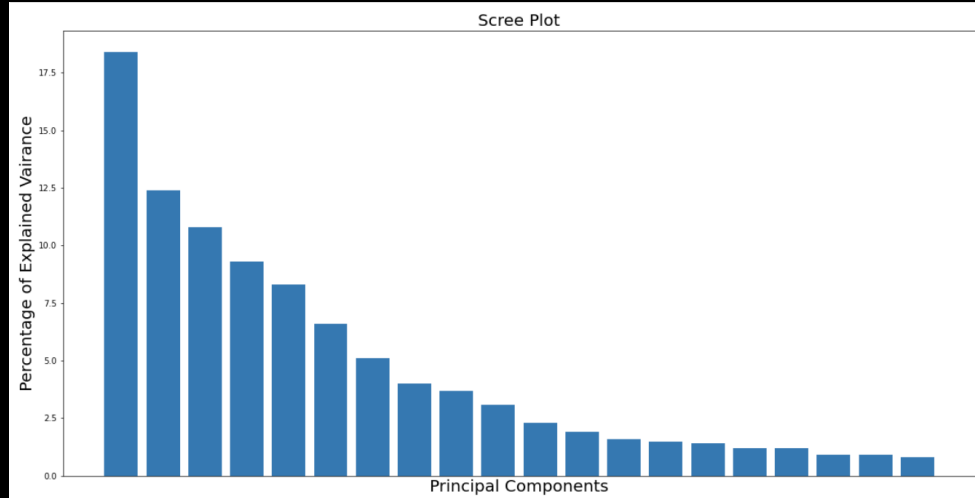
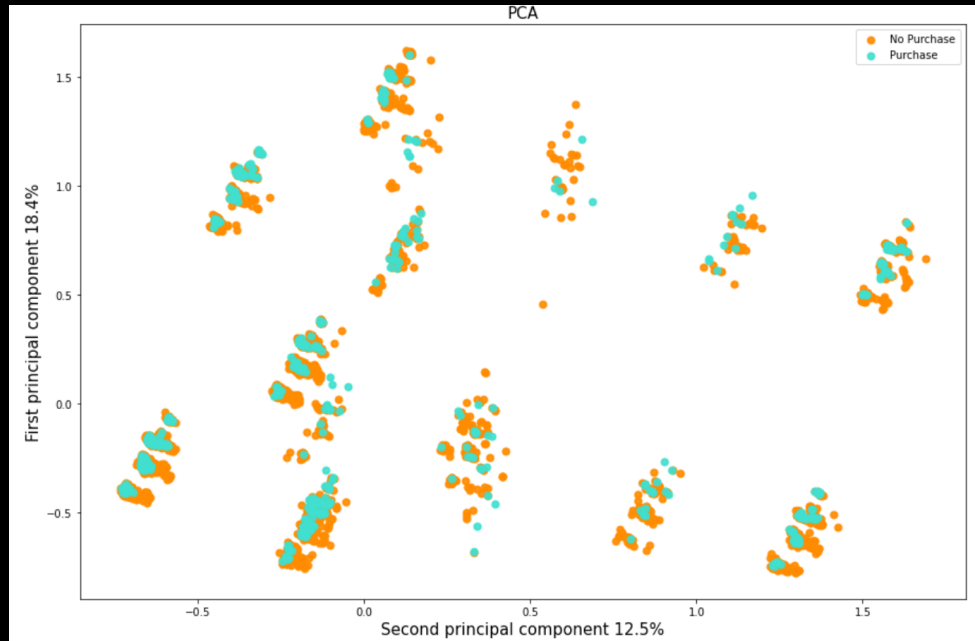
PagesValues (0,5)

ExitRates (-0,21)

BounceRates (-0,16)

ProductRelated_Duration (0,15)





Principal Components Analysis

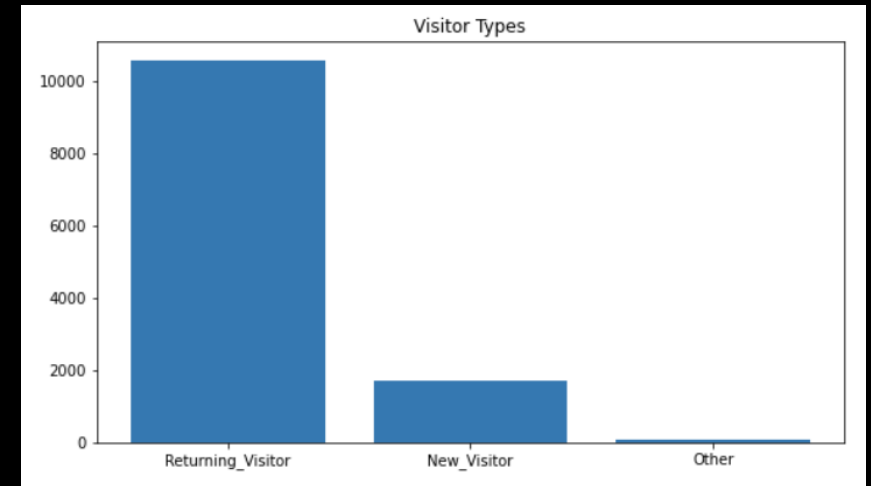
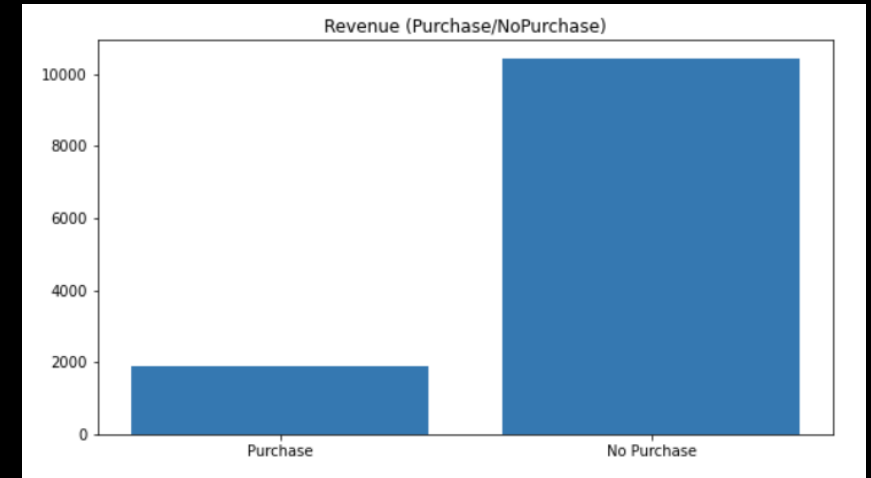
There are 12 clusters that may correspond to the 12 months.

Because the percentage of explained variance decay slowly, it is not possible to represent well our dataset in two or three dimensions.

Dataset Balance

I decided to look at the dataset balance by looking at our target (Purchased) and VisitorTypes.

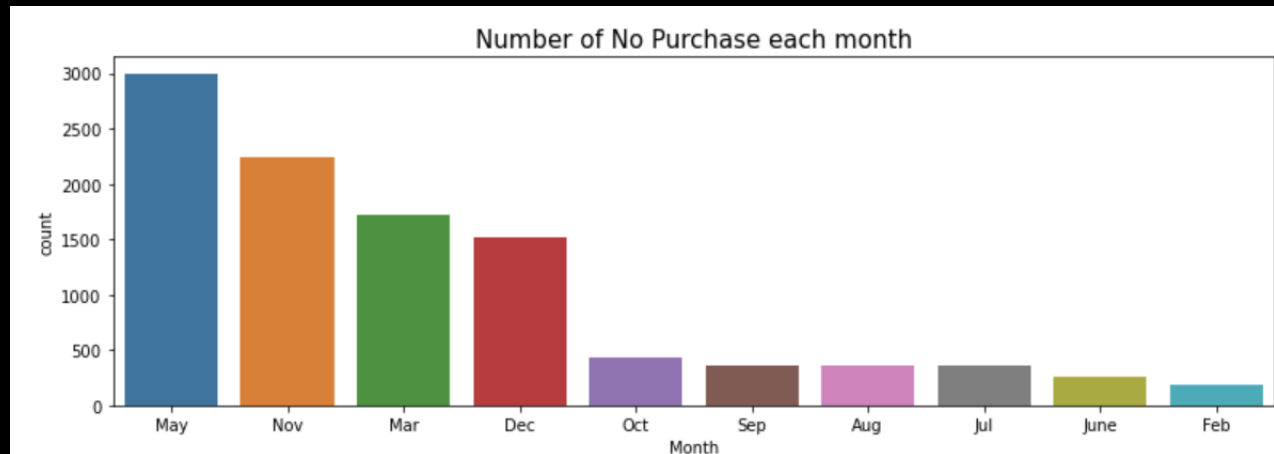
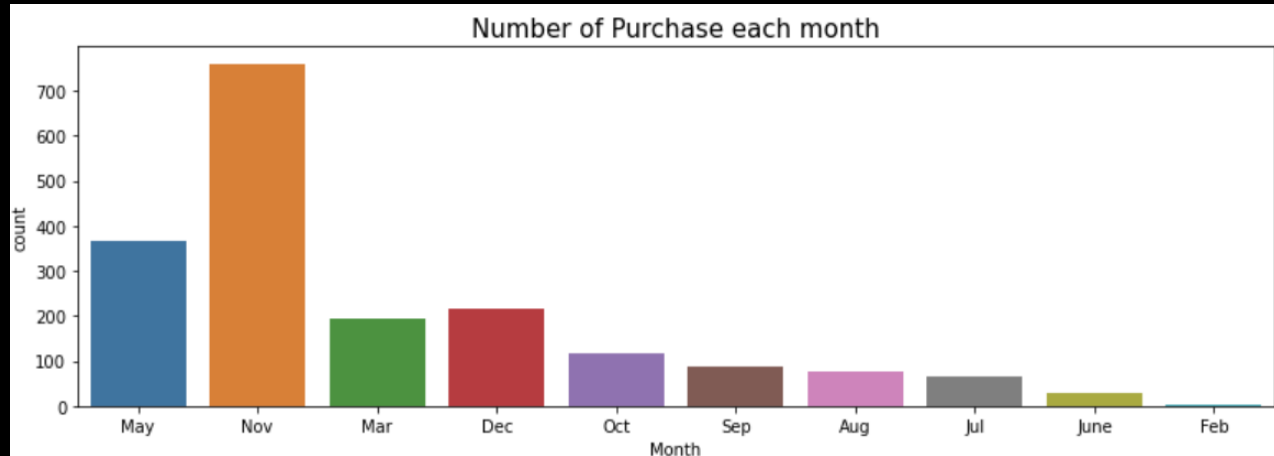
As we can see, the dataset is quite unbalanced. This means that we will need to downsample the data that leads to no purchase.



Univariate analysis

I decided to look at the dataset balance by looking at our target (Purchased) and VisitorTypes.

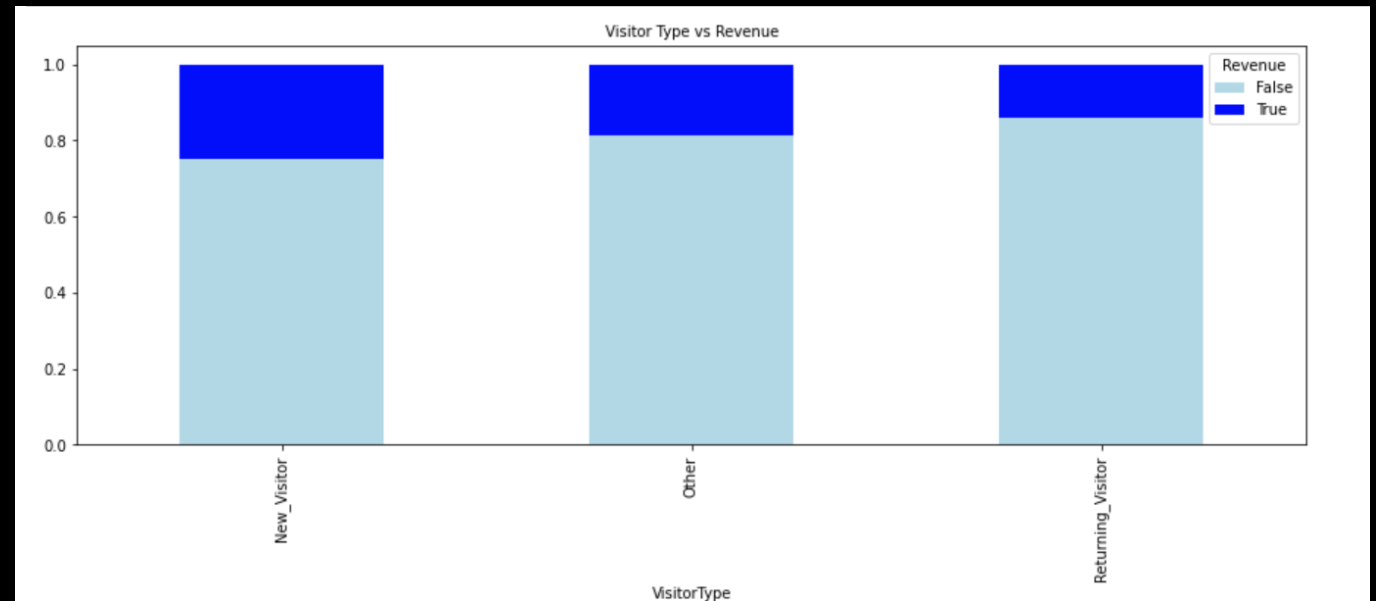
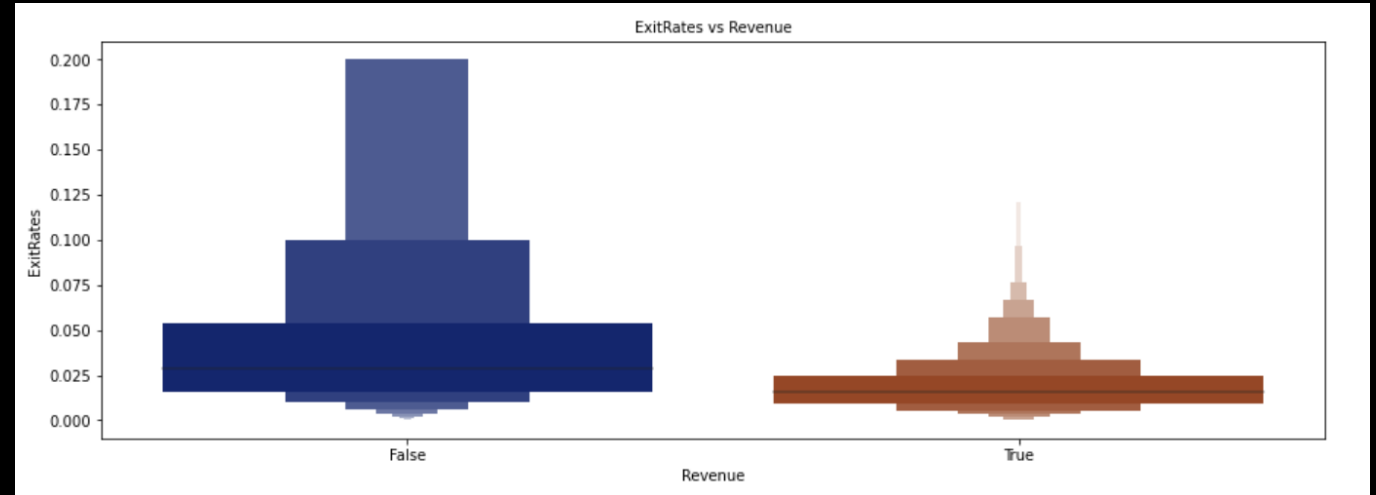
As we can see, the dataset is quite unbalanced. This means that we will need to downsample the data that leads to no purchase.



Bivariate analysis

Differences in ExitRates are not that significant by Revenue. We can note that the ExitRates are a little lower when there is a 'Purchase'.

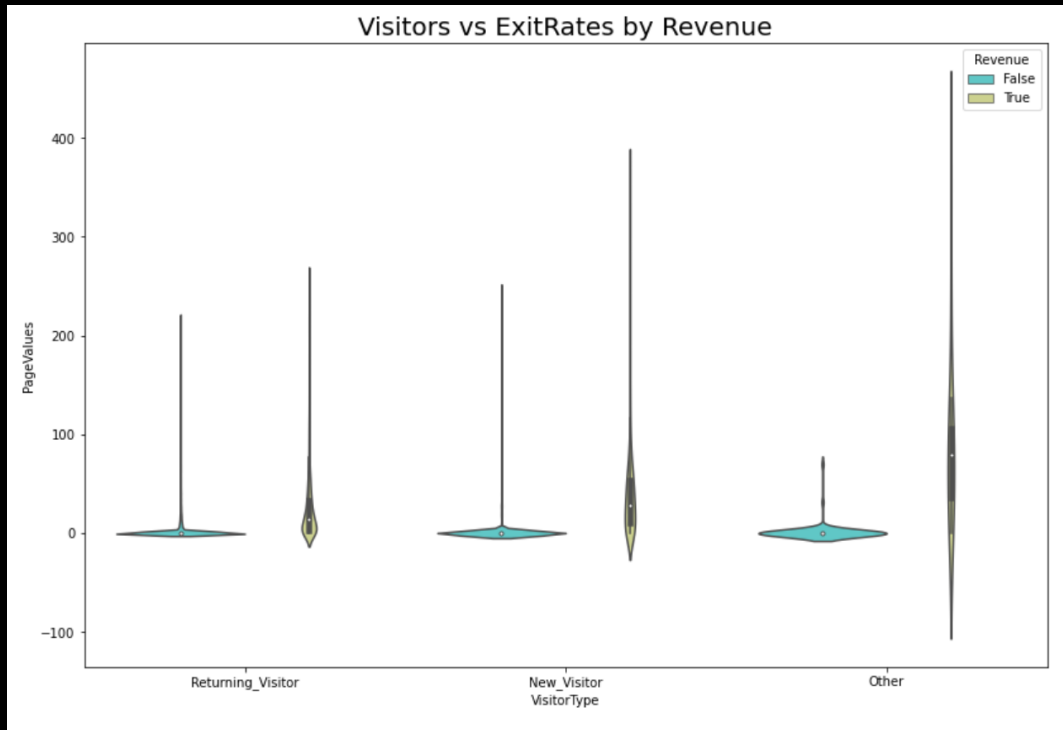
This crosstab shows us that the Visitor_Type has almost no influence on the number of Purchases.



Multivariate analysis

The violin plot is showing us the same pattern for each visitor type. Page Value for most of the entries are close to 0 when the Revenue is False. But the values are spread out when Revenue is True.

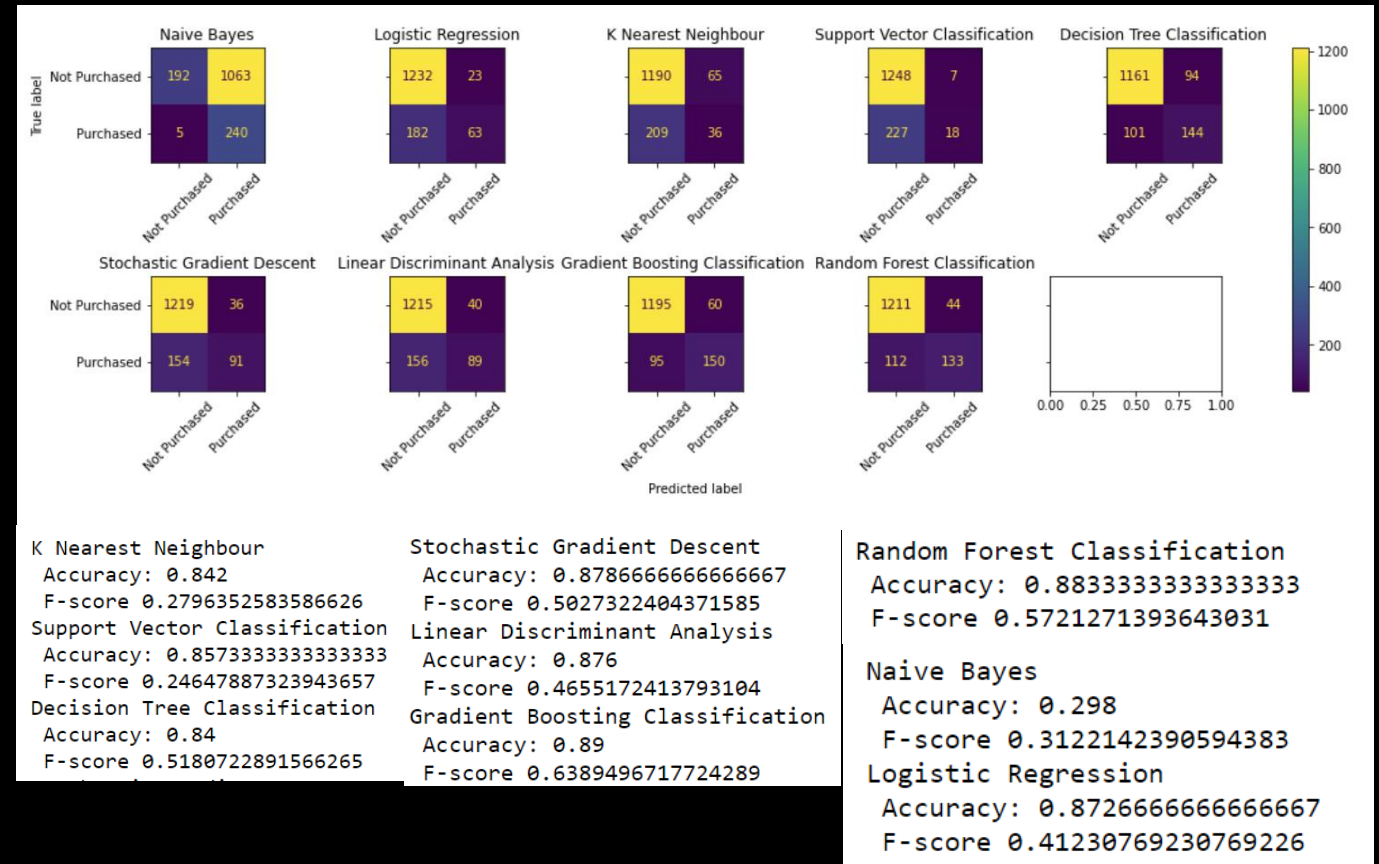
This may come from the definition of the PageValue, which gives a value close to 0 to pages which did not generate a lot of revenue.



Classification algorithm exploration

To select the best classification algorithm, I started by measuring the accuracy and F-score of each algorithm.

To have a better idea of the performance I also plot the confusion matrix of each prediction.



Hyperparameters Optimization

After a quick overview of each algorithm, I selected the four best one and tried to improve them by optimizing the hyperparameters with a Grid Search.

```
# Random forest with best parameters
gs_rf = RandomForestClassifier(bootstrap=False,
                               criterion='entropy',
                               max_depth=100,
                               max_features='sqrt',
                               min_samples_leaf=1,
                               min_samples_split=25,
                               n_estimators=800)

gs_rf.fit(X_train, y_train)
gs_rf.score(X_test, y_test)

0.9
```

```
parameters = [{
    "criterion": ["gini", "entropy"],
    "n_estimators": [200, 400, 600, 800, 1000],
    "max_features": ['auto', 'sqrt'],
    "max_depth": [5, 50, 100],
    "min_samples_split": [5, 10, 15, 20, 25, 30],
    "min_samples_leaf": [1, 5, 10, 15, 20],
    "bootstrap": [True, False]
}]
```

```
rf = GridSearchCV(RandomForestClassifier(),
                  parameters,
                  cv=5,
                  n_jobs=-1,
                  scoring='f1',
                  verbose=True)
```

```
rf.fit(X_train, y_train)
```

Fitting 5 folds for each of 3600 candidates, totalling 18000 fits

```
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done 26 tasks | elapsed: 7.0s
[Parallel(n_jobs=-1)]: Done 176 tasks | elapsed: 47.5s
[Parallel(n_jobs=-1)]: Done 426 tasks | elapsed: 2.1min
[Parallel(n_jobs=-1)]: Done 776 tasks | elapsed: 3.7min
[Parallel(n_jobs=-1)]: Done 1226 tasks | elapsed: 6.0min
[Parallel(n_jobs=-1)]: Done 1776 tasks | elapsed: 9.1min
[Parallel(n_jobs=-1)]: Done 2426 tasks | elapsed: 13.1min
[Parallel(n_jobs=-1)]: Done 3176 tasks | elapsed: 17.5min
[Parallel(n_jobs=-1)]: Done 4026 tasks | elapsed: 23.2min
[Parallel(n_jobs=-1)]: Done 4976 tasks | elapsed: 28.4min
[Parallel(n_jobs=-1)]: Done 6026 tasks | elapsed: 34.0min
[Parallel(n_jobs=-1)]: Done 7176 tasks | elapsed: 41.6min
[Parallel(n_jobs=-1)]: Done 8426 tasks | elapsed: 49.7min
[Parallel(n_jobs=-1)]: Done 9776 tasks | elapsed: 57.3min
[Parallel(n_jobs=-1)]: Done 11226 tasks | elapsed: 66.1min
[Parallel(n_jobs=-1)]: Done 12776 tasks | elapsed: 77.0min
[Parallel(n_jobs=-1)]: Done 14426 tasks | elapsed: 86.8min
[Parallel(n_jobs=-1)]: Done 16176 tasks | elapsed: 98.8min
[Parallel(n_jobs=-1)]: Done 18000 out of 18000 | elapsed: 117.3min finished
```

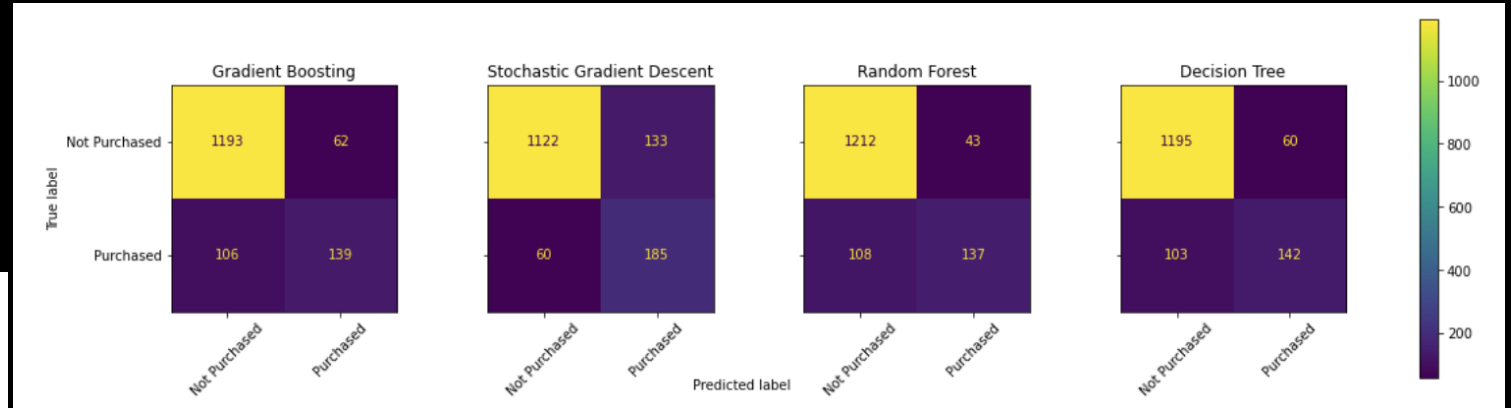
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(), n_jobs=-1,
             param_grid=[{'bootstrap': [True, False],
                           'criterion': ['gini', 'entropy'],
                           'max_depth': [5, 50, 100],
                           'max_features': ['auto', 'sqrt'],
                           'min_samples_leaf': [1, 5, 10, 15, 20],
                           'min_samples_split': [5, 10, 15, 20, 25, 30],
                           'n_estimators': [200, 400, 600, 800, 1000]}],
             scoring='f1', verbose=True)
```

```
print(rf.best_params_)
```

```
{'bootstrap': False, 'criterion': 'entropy', 'max_depth': 100, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 25, 'n_estimators': 800}
```

Final model

Accuracy : 0.8986666666666666
f1 score 0.6724137931034483



The final model is a combination of the three best models which are:

- Random Forest
- Decision Tree
- Stochastic Gradient Descent

This allowed me to achieve an accuracy of 0,899 with a F-score of 0,67.

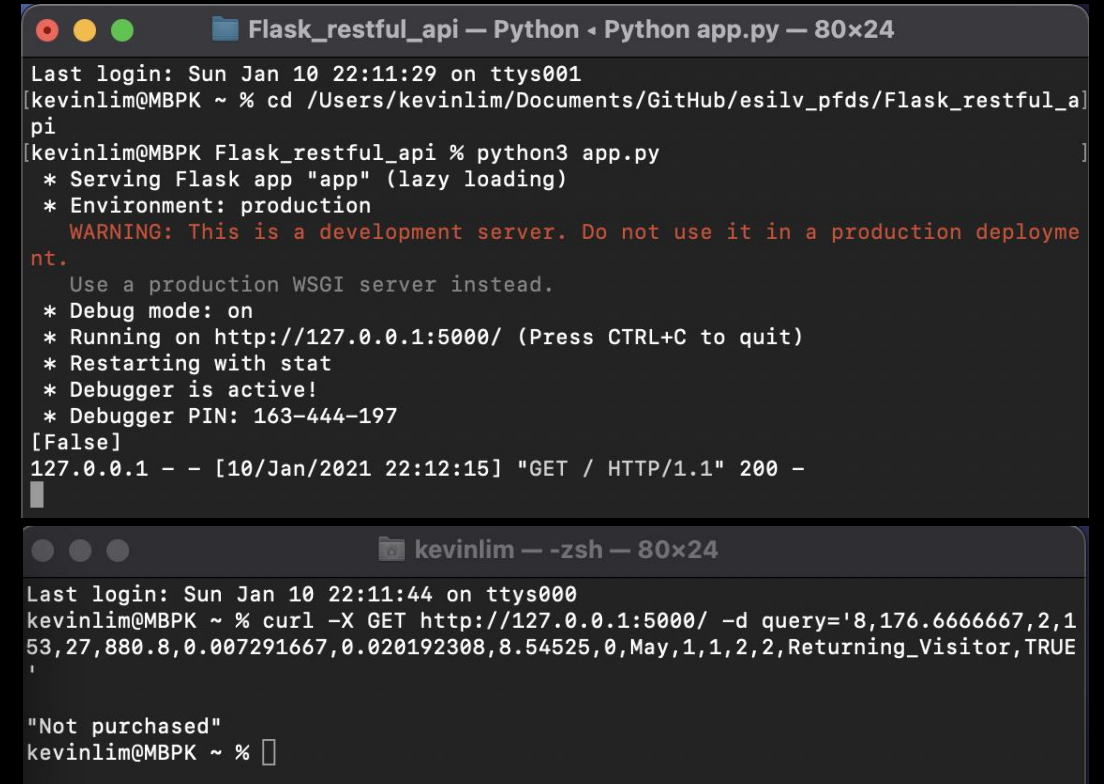
API

After the export of our model and preprocessing I built a rest api on Flask.

By inputting the values for a online shopping session, we can predict if that session leads to a Purchase or not.

Example of GET request:

```
curl -X GET http://127.0.0.1:5000/ -d query='8,176.6666667,2,153,27,880.8,0.007291667,0.020192308,8.54525,0,May,1,1,2,2,Returning_Visitor,TRUE'
```



```
Flask_restful_api — Python — Python app.py — 80x24
Last login: Sun Jan 10 22:11:29 on ttys001
[kevinlim@MBPK ~ % cd /Users/kevinlim/Documents/GitHub/esilv_pfds/Flask_restful_api
pi
[kevinlim@MBPK Flask_restful_api % python3 app.py
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 163-444-197
[False]
127.0.0.1 - - [10/Jan/2021 22:12:15] "GET / HTTP/1.1" 200 -

kevinlim — -zsh — 80x24
Last login: Sun Jan 10 22:11:44 on ttys000
kevinlim@MBPK ~ % curl -X GET http://127.0.0.1:5000/ -d query='8,176.6666667,2,153,27,880.8,0.007291667,0.020192308,8.54525,0,May,1,1,2,2,Returning_Visitor,TRUE'

"Not purchased"
kevinlim@MBPK ~ %
```