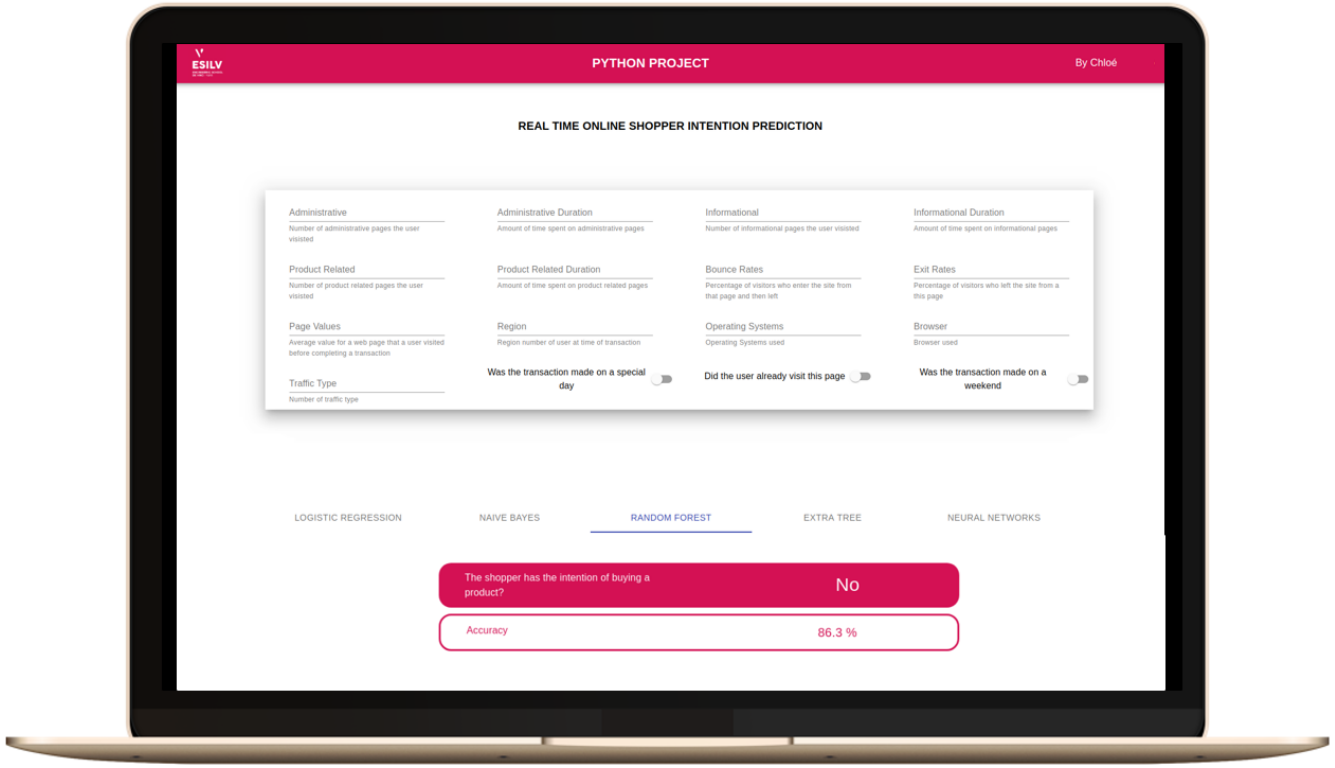
10 JANUARY 2020



Python Project

Online shopper intention predictions

CHLOE VILDE
ESILV A5
chloe.vilde@edu.devinci.fr

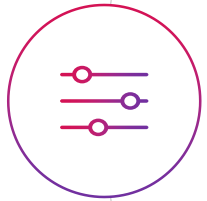
Three important steps

DATA PROCESSING

An integral step in Machine Learning as the quality of data affects the ability of our model to learn.

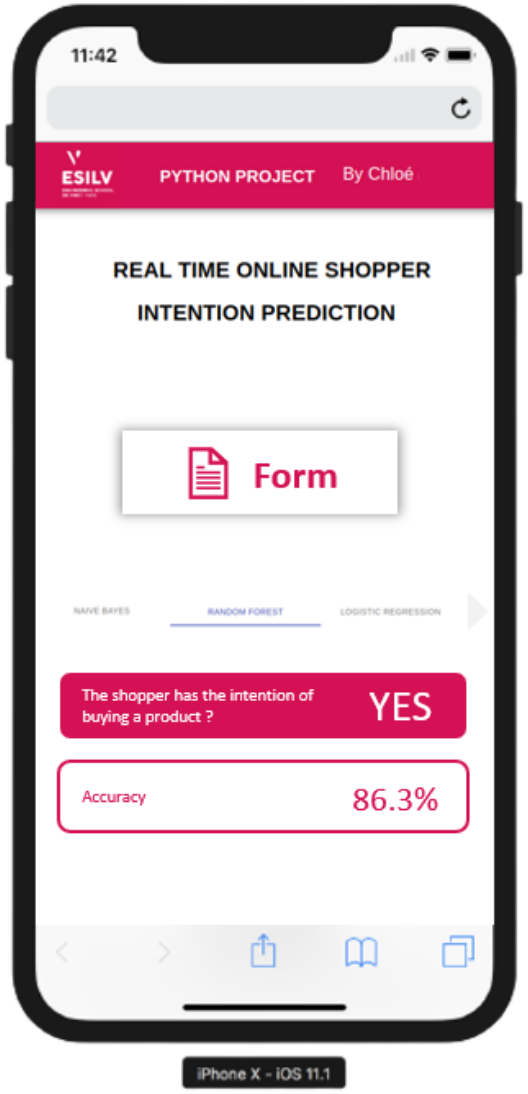
MACHINE LEARNING MODELS

Machine learning algorithms build a model based on sample data in order to make predictions or decisions without being explicitly programmed to do so



FLASK API

A web framework for Python. It provides functionality for building web applications, including managing HTTP requests and rendering templates



OUR PROJECT

The goal of this project is to predict if an online user, when entering the website, has the intention of buying a product on the site.

Using the following dataset and Flask as a web framework, we made a website able predict in **real time** if the online user has the intention of buying a product.

PYTHON PROJECT

DATA
PRE-PROCESSING

Dataset Description

Before starting to code, we wanted to fully understand the dataset we were given and its purpose.

PURPOSE
The objective of this project is to run multiple machine learning algorithms to identify the customers who will likely buy when doing online shopping.
The dataset was formed so that each session would belong to a different user in a 1-year period.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay	Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
2	0	0	0	0	1	0	0.2	0.2	0	0	Feb	1	1	1	1	Returning_Visitor	FALSE	FALSE
3	0	0	0	0	2	64	0	0.1	0	0	Feb	2	2	1	2	Returning_Visitor	FALSE	FALSE
4	0	0	0	0	1	0	0.2	0.2	0	0	Feb	4	1	9	3	Returning_Visitor	FALSE	FALSE

- Administrative: This is the number of pages of this type (administrative) that the user visisted.
- Administrative_Duration: This is the amount of time spent in this category of pages
- Informational: This is the number of pages of this type (informational) that the user visited.
- Informational_Duration: This is the amount of time spent in this category of pages.
- Product Related: This is the number of pages of this type (product related) that the user visited.
- Product Related Duration: This is the amount of time spent in this category of pages
- Bounce Rate is the percentage of visitors who enter the site from that page and then leave ("bounce") without triggering any other requests to the analytics server during that session.
- Exit Rate is calculated as for all pageviews to the page, the percentage that were the last in the session.
- Page Value represents the average value for a web page that a user visited before completing an e-commerce transaction.
- Special Day indicates the closeness of the site visiting time to a specific special day (e.g. Mother’s Day, Valentine's Day) in which the sessions are more likely to be finalized with transaction.
- The dataset also includes operating system, browser, region, traffic type, visitor type as returning or new visitor, a Boolean value indicating whether the date of the visit is weekend, and month of the year.
- Revenue : target value. Will the user end up buying a product on the website?

Dataset Info

The next step was to understand the dataset in numbers.

01 **SHAPE** : Find the number of rows and columns

```
print('Number of rows: {}'.format(df.shape[0]))
print('Number of columns: {}'.format(df.shape[1]))
```

02 **TYPE** : Find the datatypes of the columns

```
df.info()
```

03 **NULL** : Check if there was any empty cells in the dataframe. As there were none, we did not have to process these cells.

```
df.isnull().sum()
```

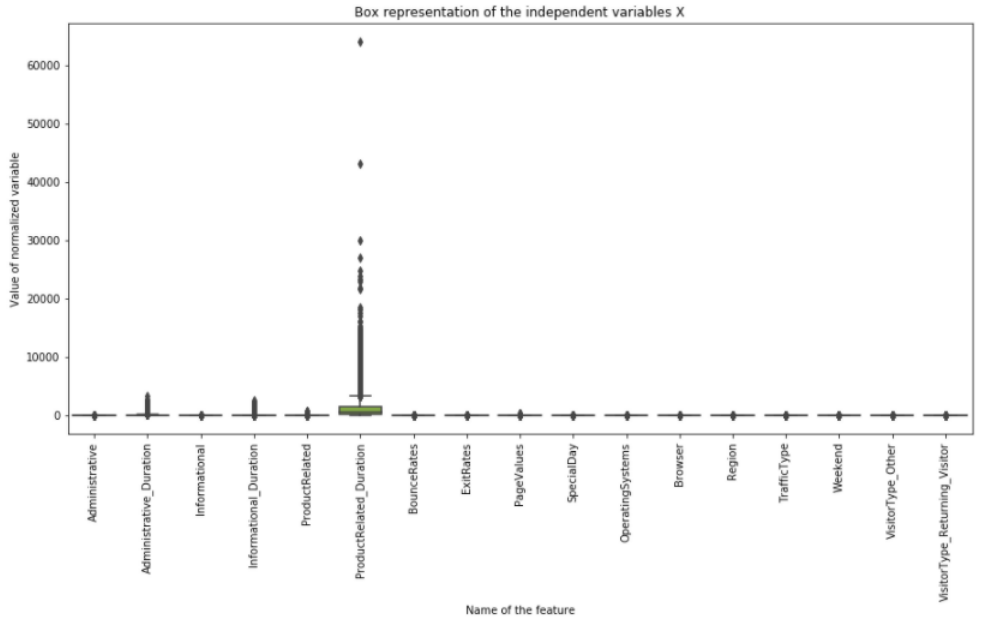
04 **CATEGORICAL** : Machine learning models require all input and output variables to be numeric. This means that any data containing categorical data, must be encoded it to numbers

```
data_dummies = pd.get_dummies(df,columns=['Month','VisitorType'],drop_first=True)
```

Data Visualization

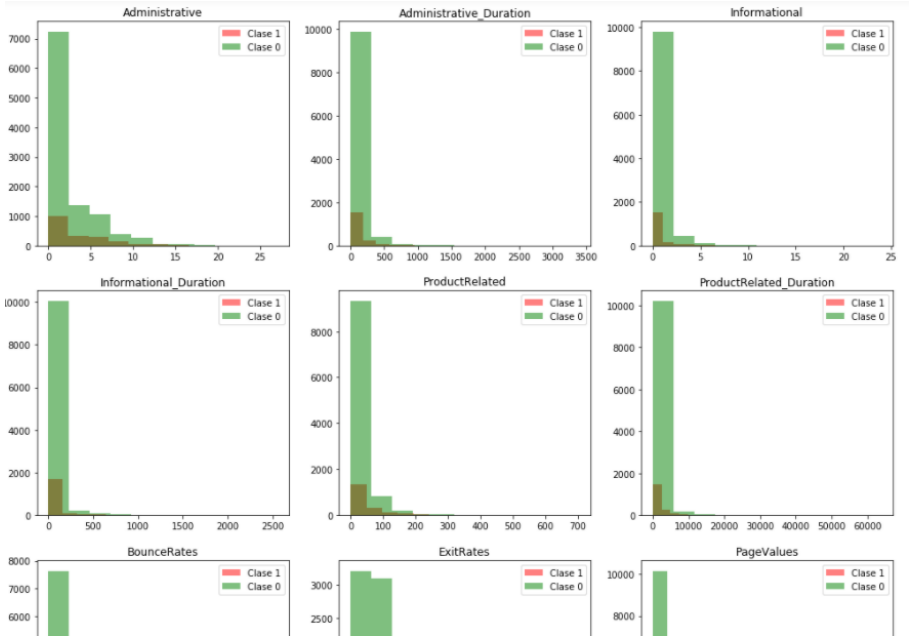
The next step was to understand the dataset visually.

01



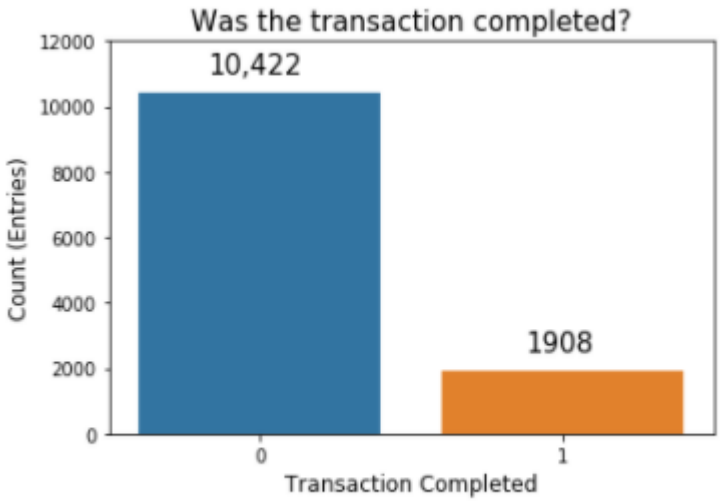
This graph shows the value of the independent variables. We can see that all the variables are concentrated on the lower part of the graph, except for ProductRelated_Duration variable. Therefore, we will standardize the data before training our Machine Learning models.

02



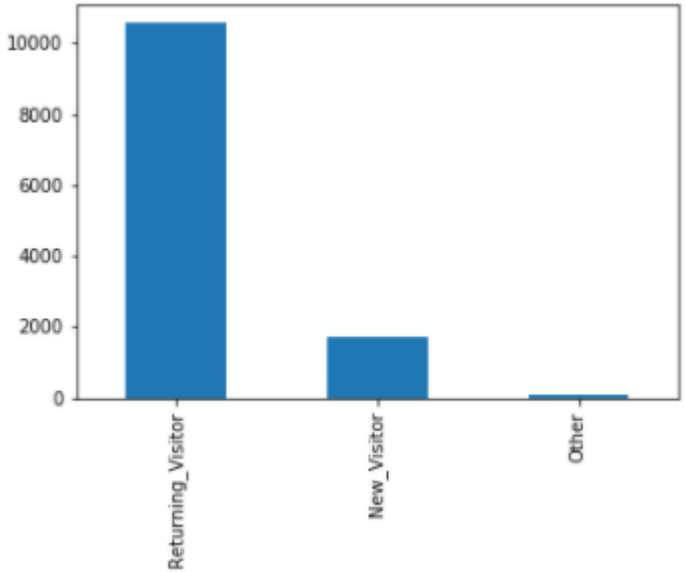
This second graph represent the independent variables into two distributions depending on their target value. Class 1 in red means that the customer has bought online. Class 0 in green is the contrary. We can see on this graph that customers who complete online purchases are evenly distributed within out data.

03



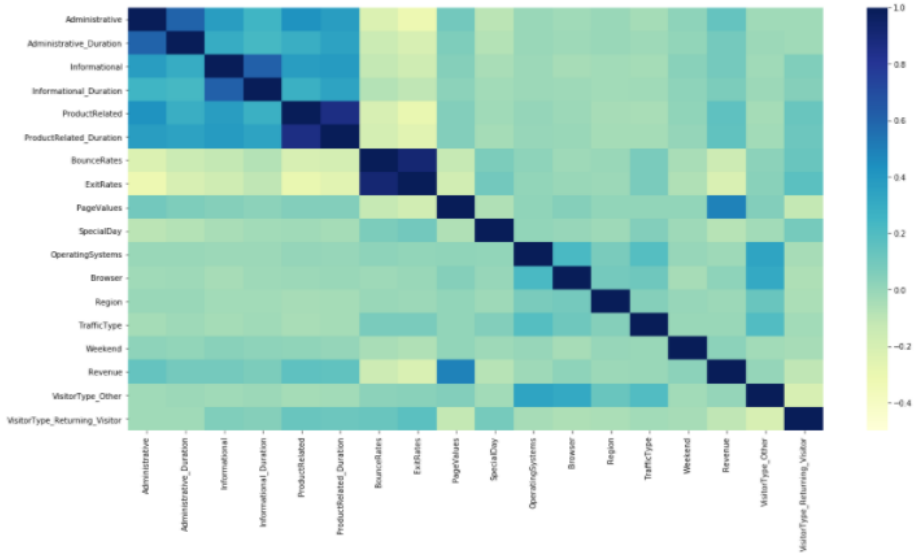
This visualization helps us understand the number of transactions completed. We were able to conclude that in 84.5% of the time, a client came online to do some shopping but did not end up buying anything.

04



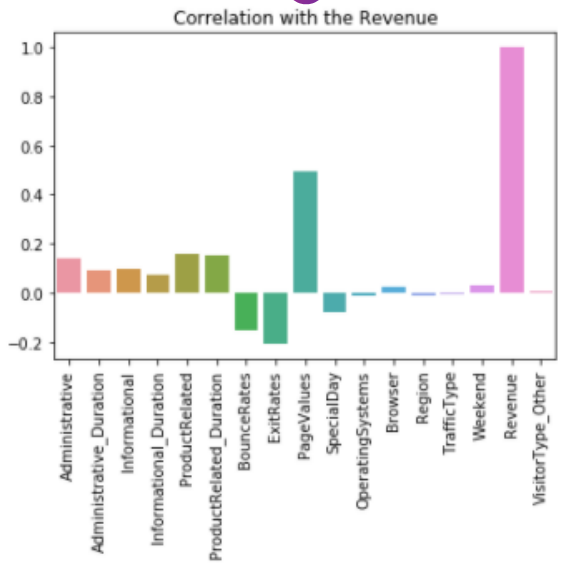
Showing the different types of visitors.
Returning_visitor is a client that had already been on the website before as a new_visitor is a client who has never been on the website before.

05



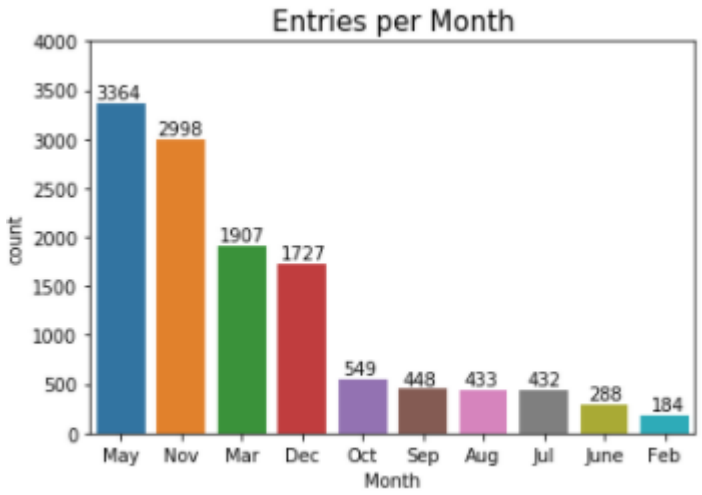
A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables.
Correlation coefficients are always values between -1 and 1, where -1 shows a perfect, linear negative correlation, and 1 shows a perfect, linear positive correlation.
In this matrix, we can see that the two different columns that have the highest correlations are ProductRelated_Duration and ProductRelated. The yellower the cell, the closer it is to the -1 value. The bluer the cell, the closer it is to the value 1.

06



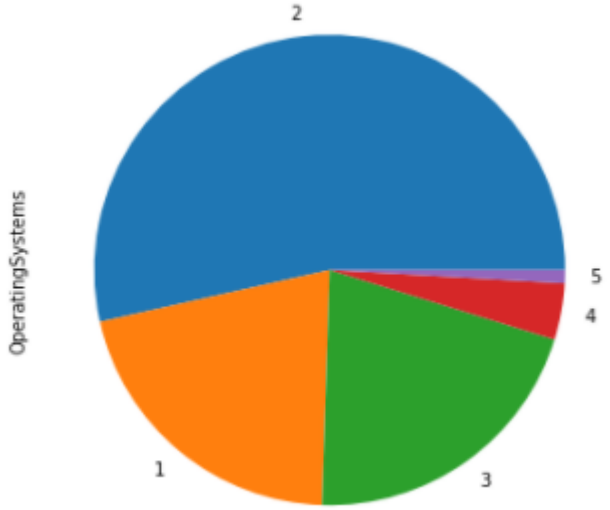
In this bar graph we compare the correlations of the independent variables with the target value (Revenue). We can wee that Page Value has the highest correlation with Revenue compared to the other variables.

07



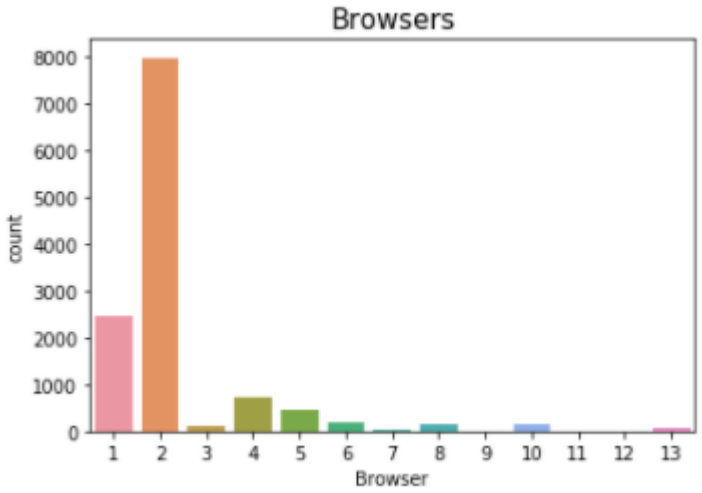
Visualization showing the amount of visits each month. We can see that May is the month where there was the most visitors and February the least.

08



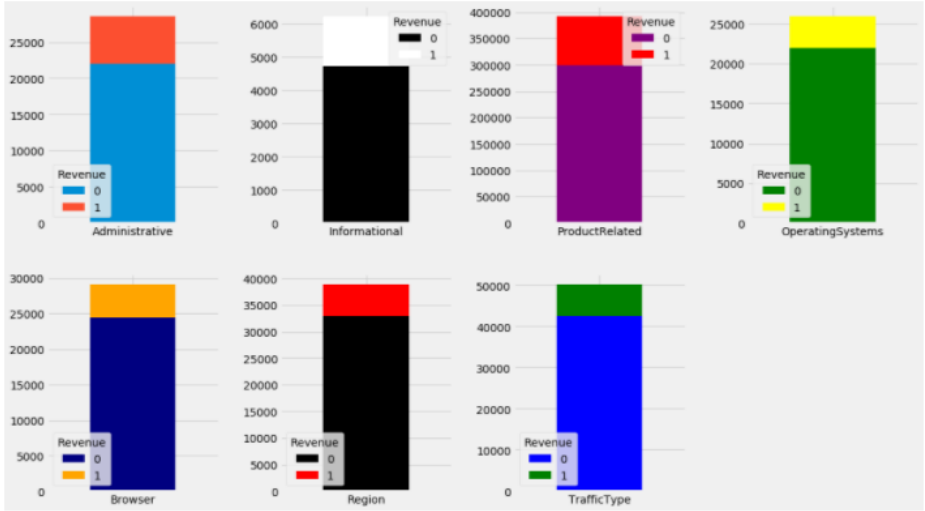
Visualization of the different types of operating systems.
We can see that most of users use operating system number 2. Operating systems can indicate users of a specifc type of computer and may portray certain user archetypes (Windows users, Mac users, Linux users).

09



Bar graph showing the different type of browsers. Browser choice is even more polarizing than Operating System. Here we see that a large majority of users use browser 2, with a smaller number of users using browser 1. All other browsers represent a small subsection of online users.

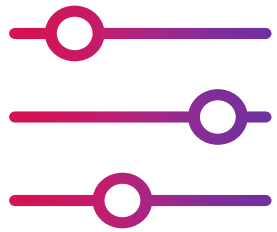
10



Visualization showing the Ratio of Revenue in each feature. We can see that the columns 'Administrative', 'Informational' and 'ProductRelated' are the most related with the revenue results.

PYTHON PROJECT

MACHINE LEARNING MODELS



Machine Learning Models

To predict whether the user has the intention of buying a product online or not, we will be using 5 different supervised algorithms

01

LOGISTIC REGRESSION

A statistical model that in its basic form uses a logistic function to model a binary dependent variables

02

NAÏVE BAYES

A probabilistic classifier, which means it predicts based on the probability of an object

03

RANDOM FOREST

Ensemble learning method which constructs a multitude of decision trees at training time

04

EXTRA TREE

Uses a simpler algorithm than Random Forest to construct the decision trees used as a members of the ensemble

05

NEURAL NETWORKS

Series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain works

Logistic Regression

1

Use the Sklearn Logistic Regression function

```
model1 = LogisticRegression(class_weight='balanced')
```

Fit the model according to the given training data

```
model1.fit(X_train, Y_train)
```

```
LogisticRegression(C=1.0, class_weight='balanced', dual=False,
                   fit_intercept=True, intercept_scaling=1, l1_ratio=None,
                   max_iter=100, multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

Predict class labels for sample in X

```
y_pred=model1.predict(X_test)
```

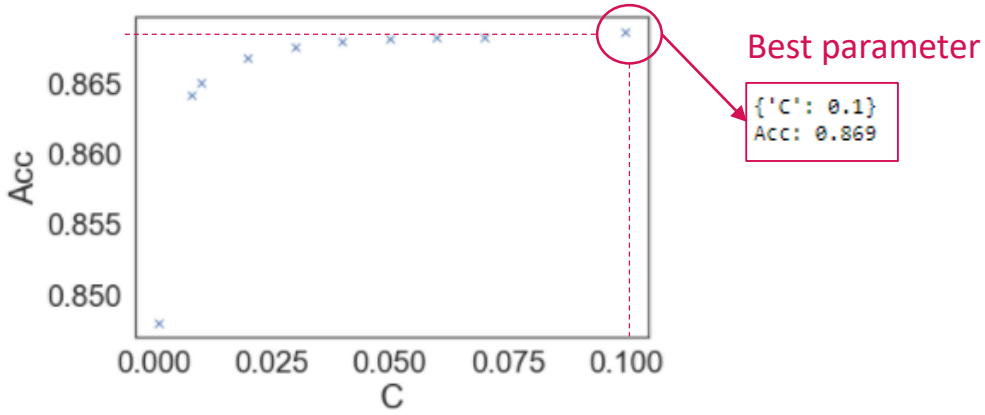
Get accuracy and recall of the model

```
print("Accuracy\t{}".format(round(metrics.accuracy_score(Y_test, y_pred),3)))
print("Recall\t{}".format(round(metrics.recall_score(Y_test, y_pred),3)))
```

```
Accuracy        0.827
Recall  0.766
```

2

Perform GridSearchCV in order to determine the optimal value for the model as the performance of a model significantly depends on the values of hyperparameters. A plot is done to visually see the results of the different hyperparameters vs their accuracies.



Best parameter

```
{'C': 0.1}
Acc: 0.869
```

3

Best threshold

We must choose the threshold to separate our results which are continuous values into a binary option of Yes or No (1 or 0) about the revenue probability.

```
LogRecAcc = bestThreshold(0.8)
```

Best threshold found

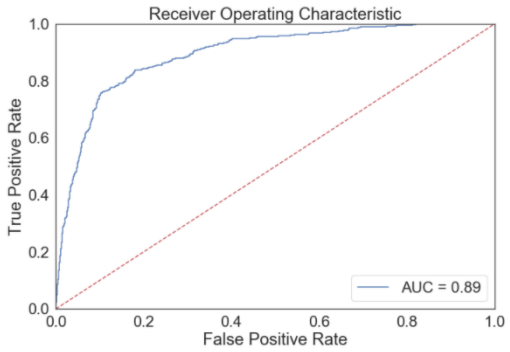
```
Confusion Matrix
 [[1982  100]
 [ 194  190]]

Accuracy        0.881
Sensitivity     0.495
Precision       0.655
```

4

ROC Curve

Graph showing the performance of a classification model at all classification thresholds

AUC (Area under the ROC Curve) : AUC ranges in value from 0 to 1. Our model has AUC = 0.89 which means it is a model whose predictions are 89 % correct.



Naïve Bayes

1

Use the Sklearn Naïve Bayes function

Fit Gaussian Naive Bayes Classier to training data

```
gnb = GaussianNB()
```

Fit the model according to the given training data

```
gnb.fit(X_train, Y_train)
```

```
GaussianNB(priors=None, var_smoothing=1e-09)
```

Make prediction using our test data and model

```
y_pred = gnb.predict(X_test)
```

Comparing our prediction to response values

```
NBCAcc = round(metrics.accuracy_score(Y_test, y_pred)*100,2)
print("Gaussian Naive Bayes model accuracy(in %):", NBCAcc)
```

```
Gaussian Naive Bayes model accuracy(in %): 80.86
```

2

Confusion Matrix

```
cm = confusion_matrix(Y_test, y_pred)
cm
```

```
array([[1858,  224],
       [ 157,  227]], dtype=int64)
```

True Positive: 1858 - correctly predicted positive class

True Negative: 227 - correctly predicted negative class

False Positive: 224 - type 1 error

False Negative: 157 - type 2 error

Random Forest

1

Use the Sklearn Random Forest function

Fit Random Forest Classifier to our Training Data

```
rfc = RandomForestClassifier(max_depth=5, random_state=2, n_estimators=750)
rfc.fit(X_train, Y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=5, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=750,
                       n_jobs=None, oob_score=False, random_state=2, verbose=0,
                       warm_start=False)
```

make prediction using our test data and model

```
y_pred_rfc = rfc.predict(X_test)
y_prob_rfc = rfc.predict_proba(X_test)[:, 1]
```

Comparing our prediciton to response values

```
RFCAcc =  round(metrics.accuracy_score(Y_test, y_pred_rfc)*100,2)
print('Random Forest Classifier model accuracy(in %):', RFCAcc)
```

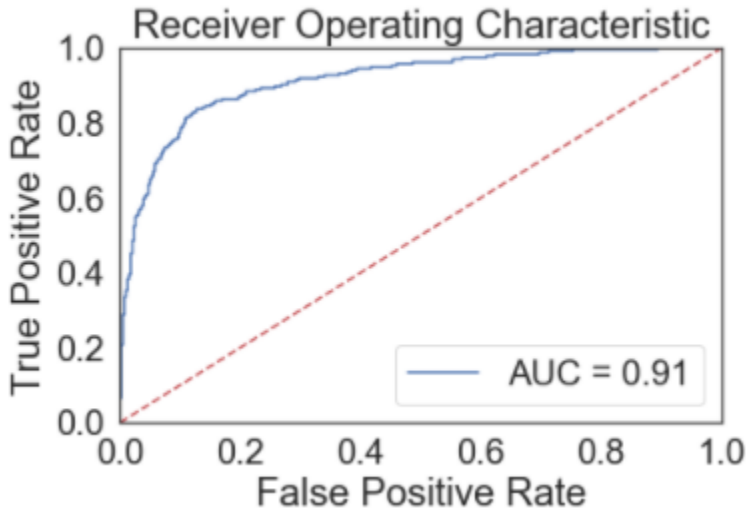
```
Random Forest Classifier model accuracy(in %): 90.27
```

2

ROC Curve

Graph showing the performance of a classification model at all classification thresholds

AUC (Area under the ROC Curve) :
AUC ranges in value from 0 to 1. Our model has AUC = 0.91 which means it is a model whose predictions are 91 % correct.



Extra Tree

1

Use the Sklearn Extra Tree Forest function

Fit Extra Trees Classifier to our Training Data

```
etc = ExtraTreesClassifier(random_state=2, n_estimators=1000)
etc.fit(X_train, Y_train)
```

```
ExtraTreesClassifier(bootstrap=False, ccp_alpha=0.0, class_weight=None,
                     criterion='gini', max_depth=None, max_features='auto',
                     max_leaf_nodes=None, max_samples=None,
                     min_impurity_decrease=0.0, min_impurity_split=None,
                     min_samples_leaf=1, min_samples_split=2,
                     min_weight_fraction_leaf=0.0, n_estimators=1000,
                     n_jobs=None, oob_score=False, random_state=2, verbose=0,
                     warm_start=False)
```

make prediction using our test data and model

```
y_pred_etc = etc.predict(X_test)
y_prob_etc = etc.predict_proba(X_test)[:, 1]
```

Comparing our prediciton to response values

```
ETCAcc = round(metrics.accuracy_score(Y_test, y_pred_etc)*100,2)
print('Extra Trees Classifier model accuracy(in %):', ETCAcc)
```

```
Extra Trees Classifier model accuracy(in %): 89.46
```

2

Confusion Matrix

```
cm = confusion_matrix(Y_test, y_pred)
cm
```

```
array([[1858,  224],
       [ 157,  227]], dtype=int64)
```

True Positive: 1858 - correctly predicted positive class

True Negative: 227 - correctly predicted negative class

False Positive: 224 - type 1 error

False Negative: 157 - type 2 error

Neural Network

1

Use the Sklearn Extra Tree Forest function

```
model = Sequential([
    Flatten(input_shape=(17,)),
    Dense(32, activation='relu'),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid'),
])
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
hist = model.fit(X_train, Y_train, batch_size=1)
```

```
9864/9864 [==============================] - 8s 822us/step - loss: 0.2550 - acc
uracy: 0.8972
```

```
hist.history['accuracy'][0] * 100
```

```
89.72019553184509
```

```
test_loss, test_acc = model.evaluate(X_test, Y_test)
NNCAcc = test_acc*100
print('Test accuracy:', NNCAcc)
```

```
78/78 [==============================] - 0s 953us/step - loss: 0.2519 - accurac
y: 0.8970
Test accuracy: 89.6999180316925
```

```
y_pred = model.predict(X_test)
```

```
y_pred = (y_pred > 0.5)
```

2

Confusion Matrix

```
cm = confusion_matrix(Y_test, y_pred)
cm
```

```
array([[1992,   90],
       [ 164,  220]], dtype=int64)
```

True Positive: 2013 - correctly predicted positive class

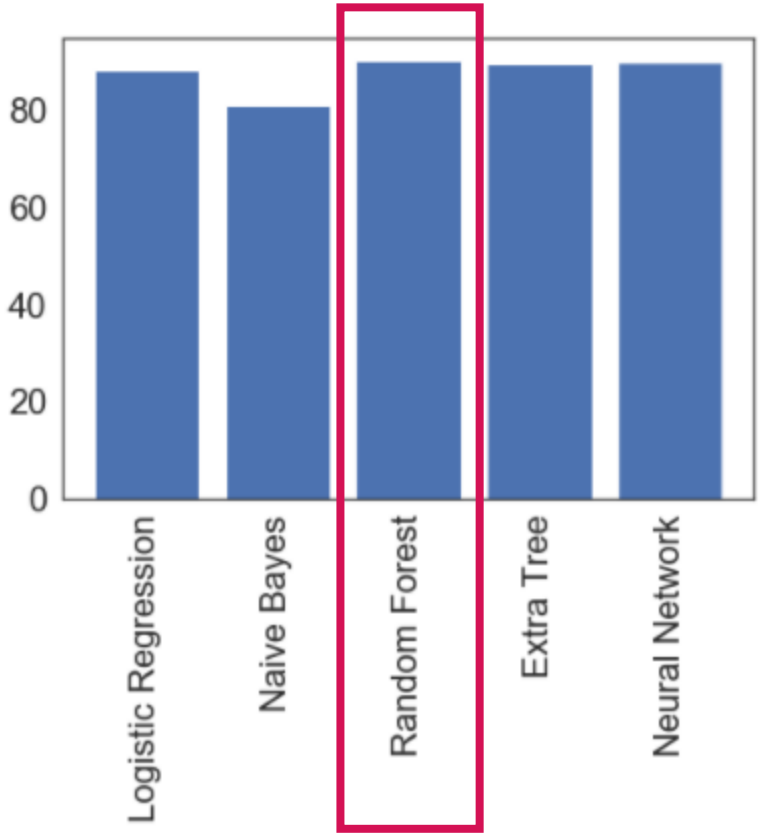
True Negative: 2020 - correctly predicted negative class

False Positive: 69 - type 1 error

False Negative: 182 - type 2 error

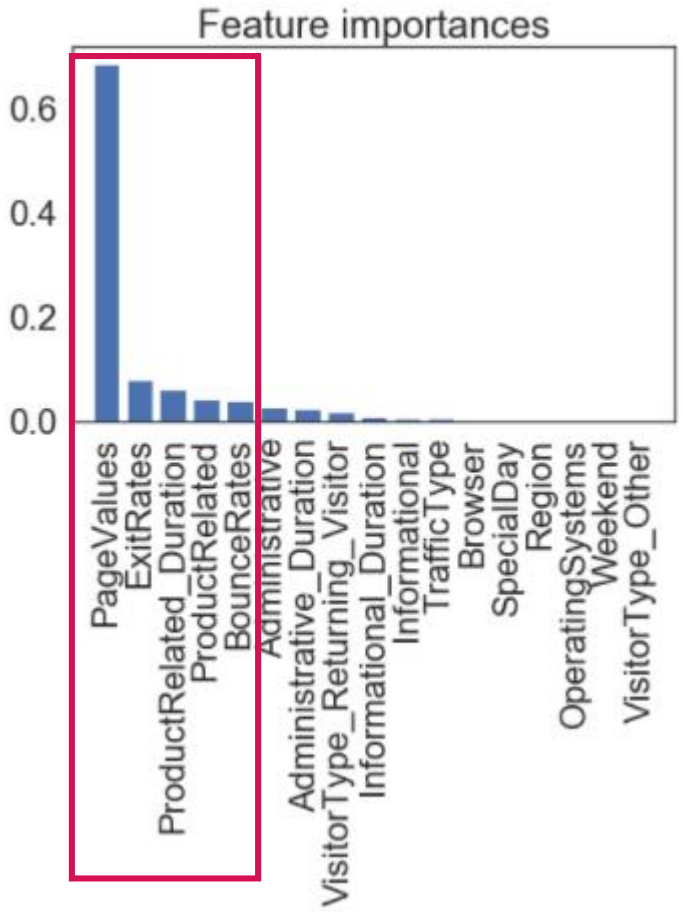
Accuracy of the models

#1 Accuracy : 90.27%



Feature Importance

To see if we can improve our model, let us track the feature importance of each of our features to see which features matter to the outcome of the prediction.
We want to see which variables contribute the most to the model



	Importance
PageValues	0.683056
ExitRates	0.079474
ProductRelated_Duration	0.061220
ProductRelated	0.041496
BounceRates	0.039001
Administrative	0.027073
Administrative_Duration	0.022268
VisitorType_Returning_Visitor	0.017502
Informational_Duration	0.007521
Informational	0.005535
TrafficType	0.004882
Browser	0.002735
SpecialDay	0.002455
Region	0.002431
OperatingSystems	0.002234
Weekend	0.000784
VisitorType_Other	0.000332

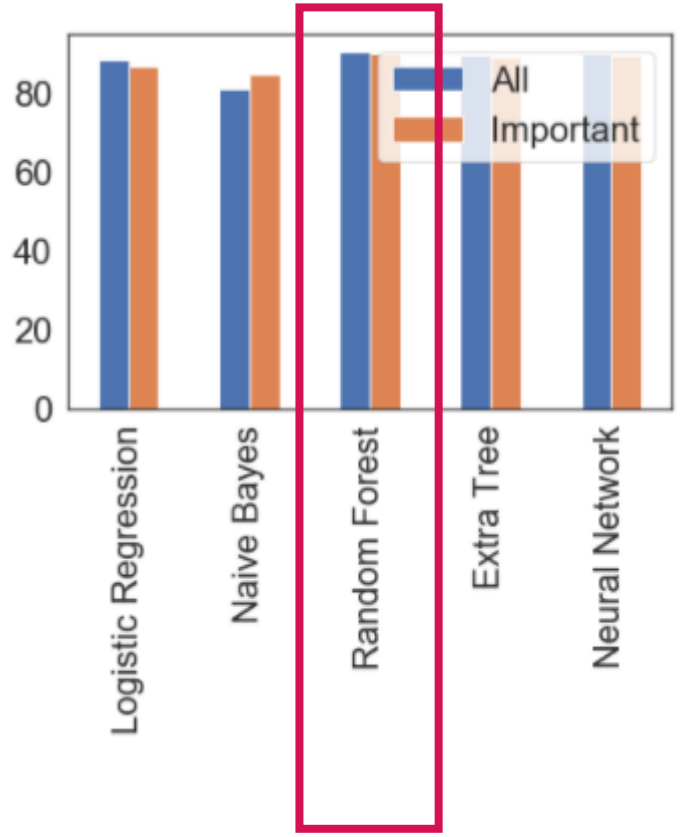
5 most important features to the model

We want to simplify our model to only use features that may heavily contribute to our classification. Using our feature importance chart, we will take the top 5 most impactful features: PageValues, ExitRates, ProductRelated_Duration, BounceRates, ProductRelate (see next slide for results).

Accuracy of the models with only important features

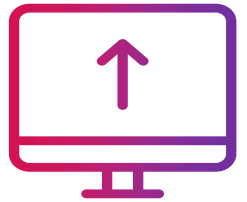
- Keeping only the first 5 important features has only increased the accuracy of the Naïve Bayes model.
- The other models have lower accuracies.
- Random Forest still has the best accuracy.

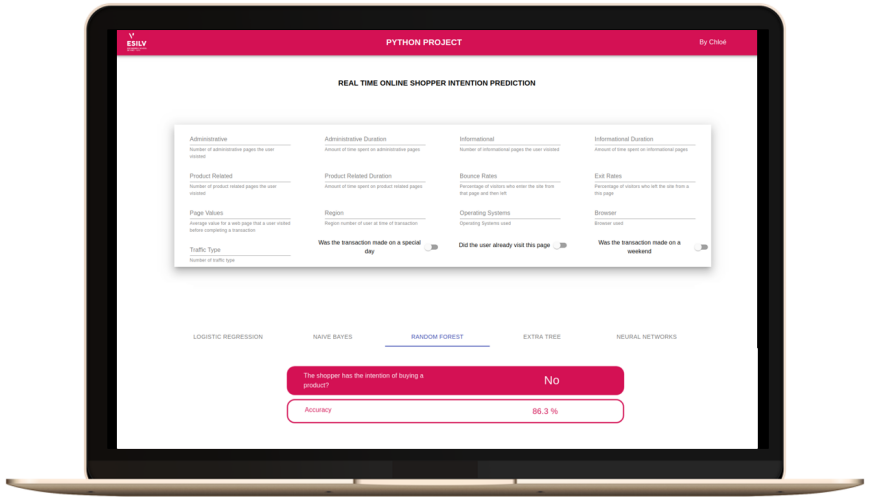
#1 Accuracy : 89.94%



PYTHON PROJECT

FLASK
API





ReactJS

Frontend

ReactJS will be used for the frontend for building the user interface and interact with the API.

Flask

Backend

Flask will act as the API service to do some python specific tasks and send the results back to the frontend.

API Endpoints

/logisticRegression POST

Uses the trained Logistic Regression model and predicts the output of the user's input.

/naiveBayes POST

Uses the trained Naïve Bayes regression model and predicts the output of the user's input.

/randomForest POST

Uses the trained Random Forest model and predicts the output of the user's input.

/extraTree POST

Uses the trained Extra Tree model and predicts the output of the user's input.

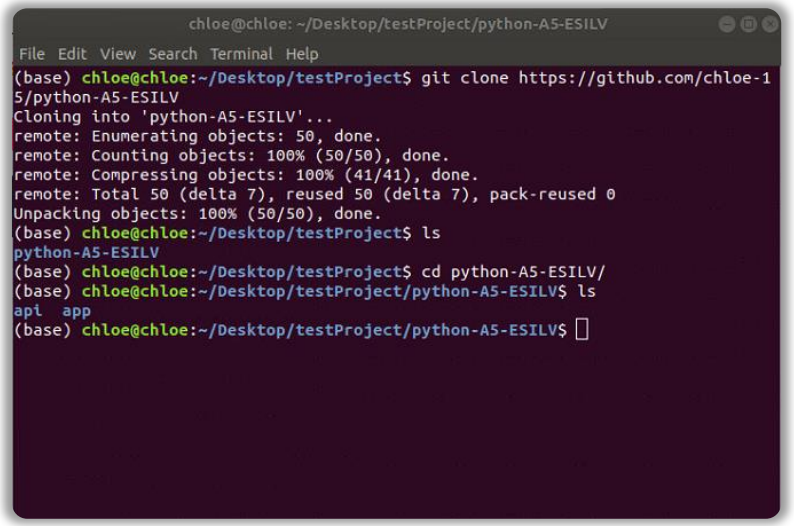
/neuralNetwork POST

Uses the trained Neural Network model and predicts the output of the user's input.

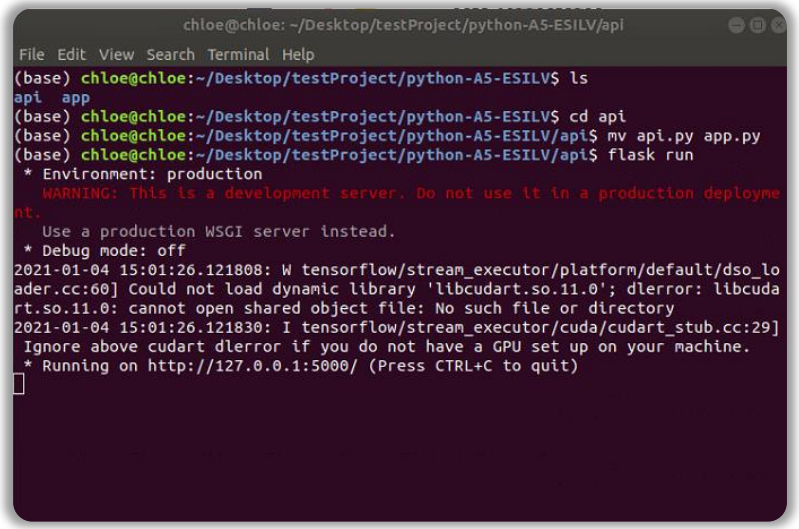
/accuracies GET

Calculates the accuracies as a percentage of each model.

Test it!

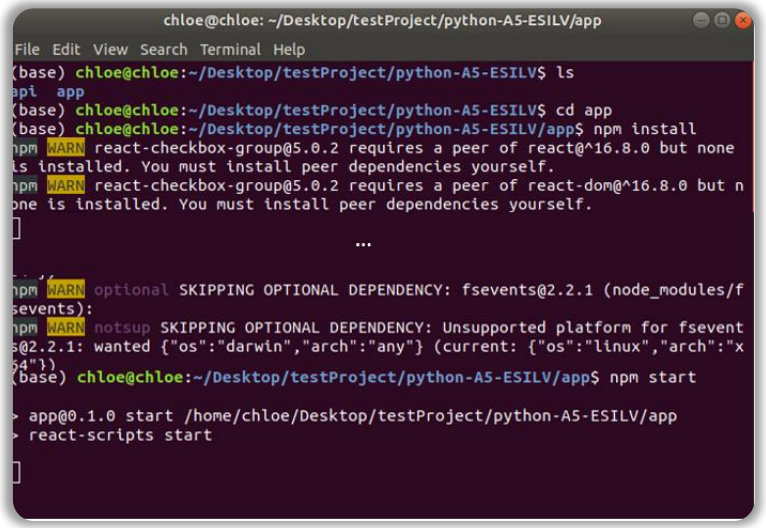


Clone the project from Github using git



Start the Flask API

API is available on your port 5000



Launch ReactJS app

Automatically opens a browser with the website