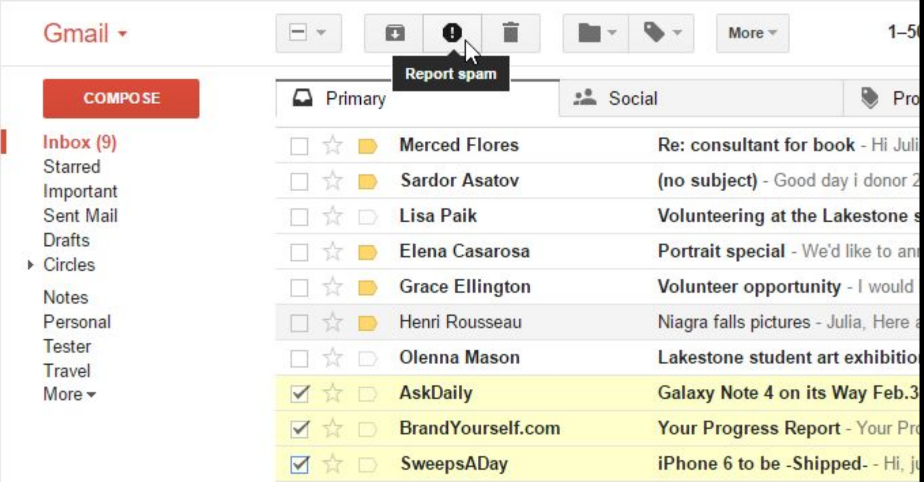
PYTHON FOR DATA ANALYSIS MASTER 1 1ST SEMESTER - PROJECT

Chloé DEPERTHES, Victoria GAUTHIER, Rémi KALBE

Data Visualisation and Prediction



WHAT IS SPAM ?

- Spam is unsolicited and unwanted junk email sent out in bulk to indiscriminate recipient lists, either for publicity or as an attempt to scam people.

- This menace increases on a yearly basis: it is responsible for over 77% of the whole global e-mail traffic and has resulted in financial loss to many users who have fallen for the scams.

CONTENT BASED FILTERING TECHNIQUE

1. The Solution: Filters

Big tech companies such as Google (for Gmail) and Apple (for iMessage) have developed an intense need for the development of machine learning-based spam filters.

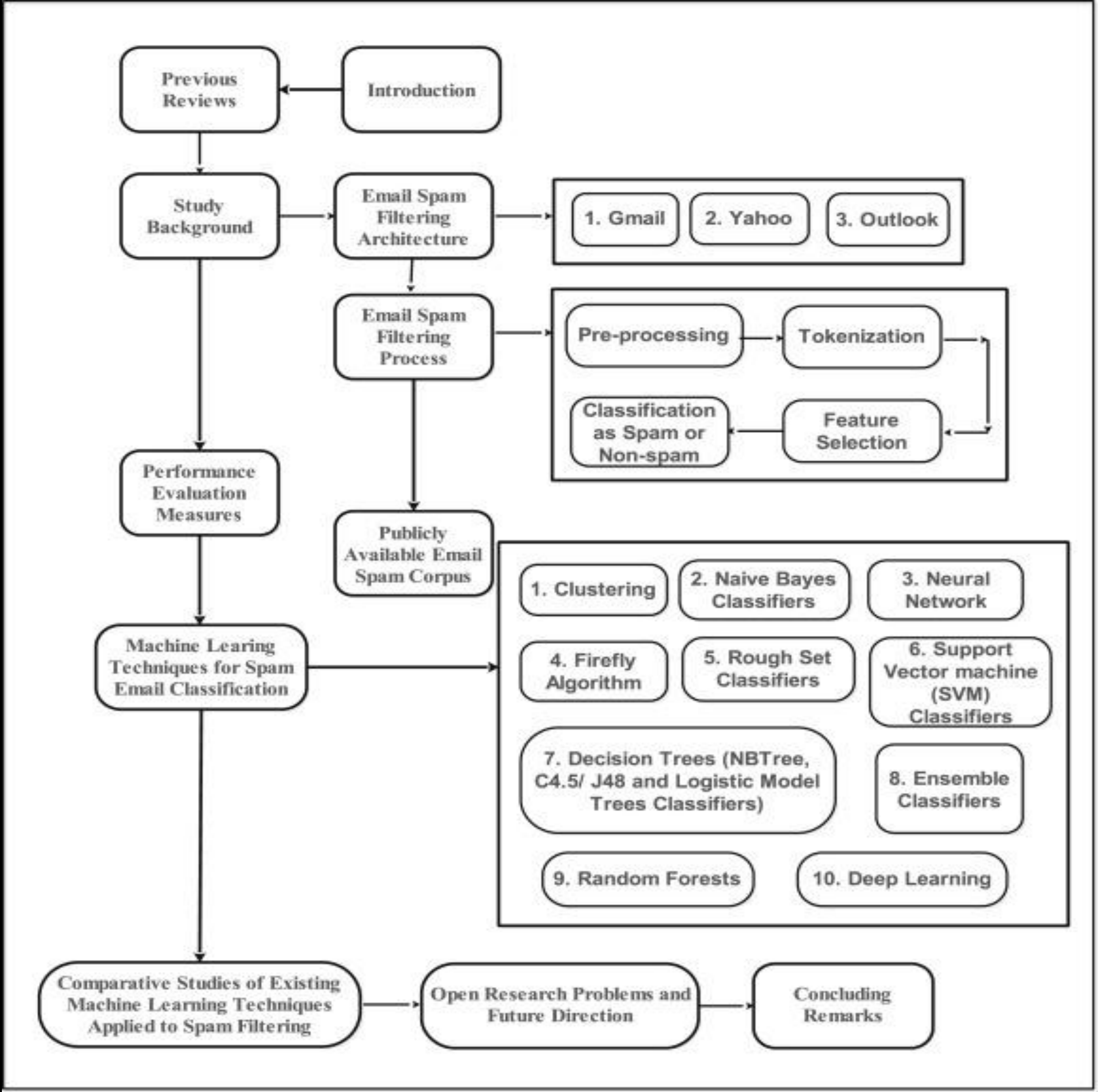
Thought there are several email spam filtering methods in existence, the most widely used is the **Content Based Filtering Technique**.

2. Content-Based :

This method analyses **words, the occurrence, and distributions of words and phrases** in the content of emails and used then use generated rules to filter the incoming email spams

GOAL OF THE PROJECT :

How can we recognize spam, and what would be the best method for doing so?



SPAMBASE DATASET CHARACTERISTICS

Data Set Characteristics:	Multivariate	**Number of Instances:**	4601	**Area:**	Computer
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	57	**Date Donated**	1999-07-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	709002

Mail origin
Spam mail : individuals who had filed spam.
Non-spam mail : filed work and personal e-mails (indicated by ‘George’ and ‘650’)

Attribute	Definition
48 continuous real [0,100] attributes of type word_freq_WORD	percentage of words in the e-mail that match WORD
6 continuous real [0,100] attributes of type char_freq_CHAR]	percentage of characters in the e-mail that match CHAR
1 continuous real [1,...] attribute of type capital_run_length_average	average length of uninterrupted sequences of capital letters
1 continuous integer [1,...] attribute of type capital_run_length_longest	length of longest uninterrupted sequence of capital letters
1 continuous integer [1,...] attribute of type capital_run_length_total	total number of capital letters in the e-mail
1 nominal {0,1} class attribute of type spam	denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

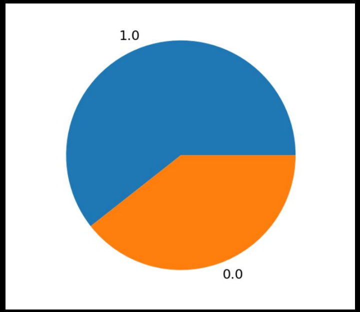
VISUALIZATION : LINK BETWEEN DATA AND TARGET

Raw Data : Spam/Not spam

Class columns

0 = Not spam (2788)

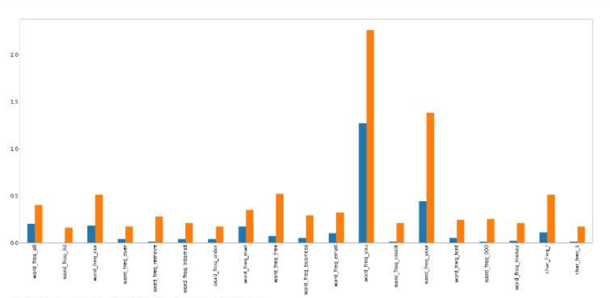
1 = Spam (1812)



Does the average of the explanatory values change, when grouped by spam/not spam ?

'our', 'free', 'you, 'your' and '!' occurrences increase when mail is considered spam.

	0.0	1.0
word_freq_you	1.27	2.26
word_freq_your	0.44	1.38
word_freq_free	0.07	0.52
word_freq_our	0.18	0.51
char_freq_!	0.11	0.51



<Figure size 600x600 with 0 Axes>

What does this mean ?

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

- Bayes thm : Probability of these words ONCE mail considered spam.
- Interpretation:

'Our', 'you', 'your' ⟶ Marketing tactic, get the recipients to trust by feeling concerned. (cf. Jupyter, use of 'George') COMMON WORDS, interesting because wouldn't be counted as 'spam features' generally (that's the .)

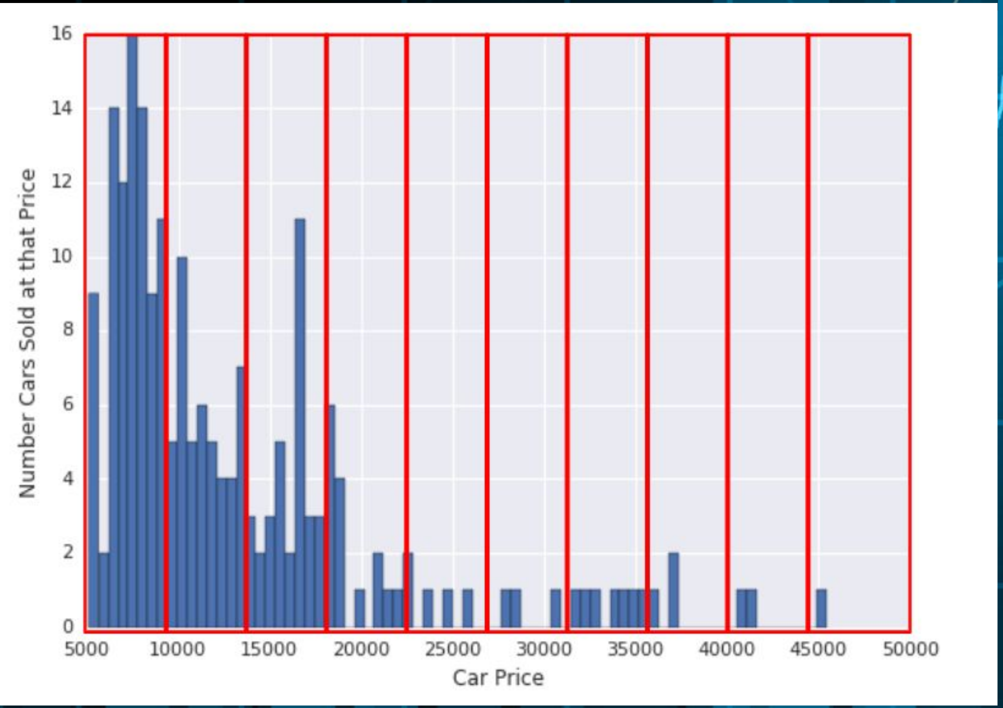
'!' ⟶ Use of urgency for marketing purposes

'Free' ⟶ obvious reasons

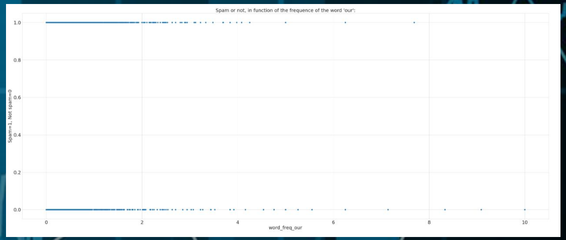
Link between Variables and Target

- Aggregation : necessary when there are more features than records
- Reduce minor observation errors (here, by averaging)
- We are ROUNDING the data
- Intervals
- Probabilities

Binning/Bucketing and Aggregating

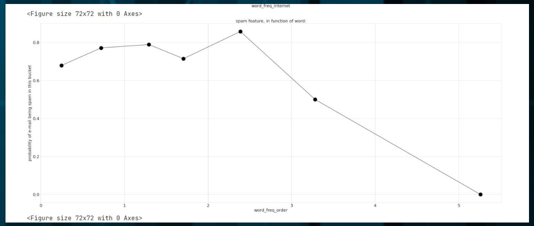


Scatter plotting raw explanatory variable column data / raw Class column data :



VS

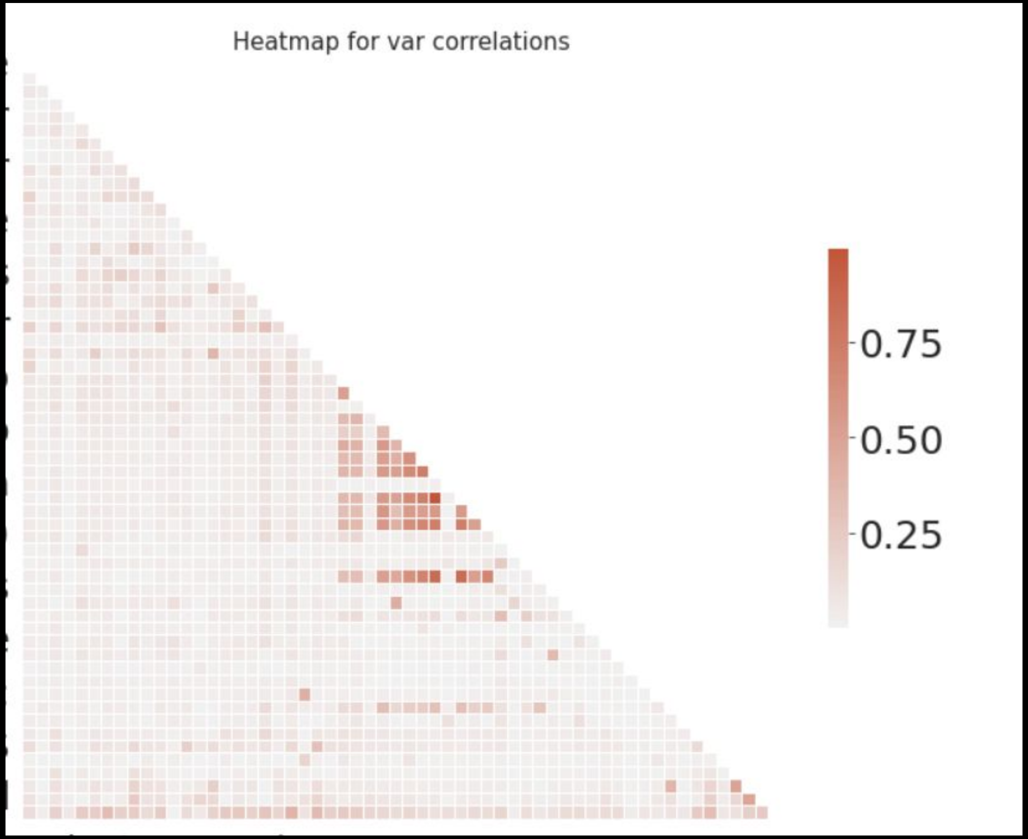
Scatter plotting Binned / Bucketed data, aggregation by average. word_freq_order :



Variable Correlation

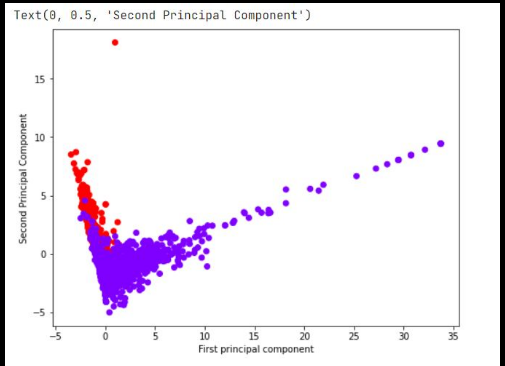
Plotted heatmap

- Correlation/Dependance: statistical relationships
- Overfitting DETECTION method (precedes the overfitting AVOIDANCE method : PCA)



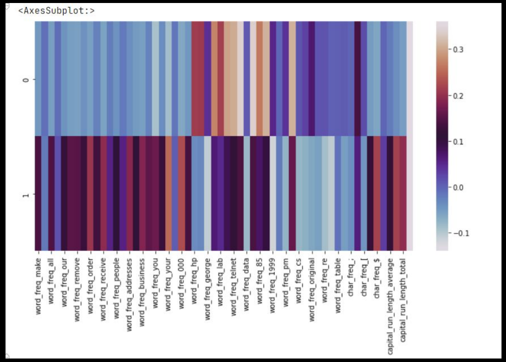
PRINCIPAL COMPONENT ANALYSIS FOR REDUCING DIMENSIONALITY OF DATA

- Principal components capture the most variation in a dataset
- Two cluster configuration, variability in different directions
- Very separable data



Two clusters: is_spam =0 or =1

- This heatmap and the color bar represent the correlation between the various feature and the principal component itself :
-

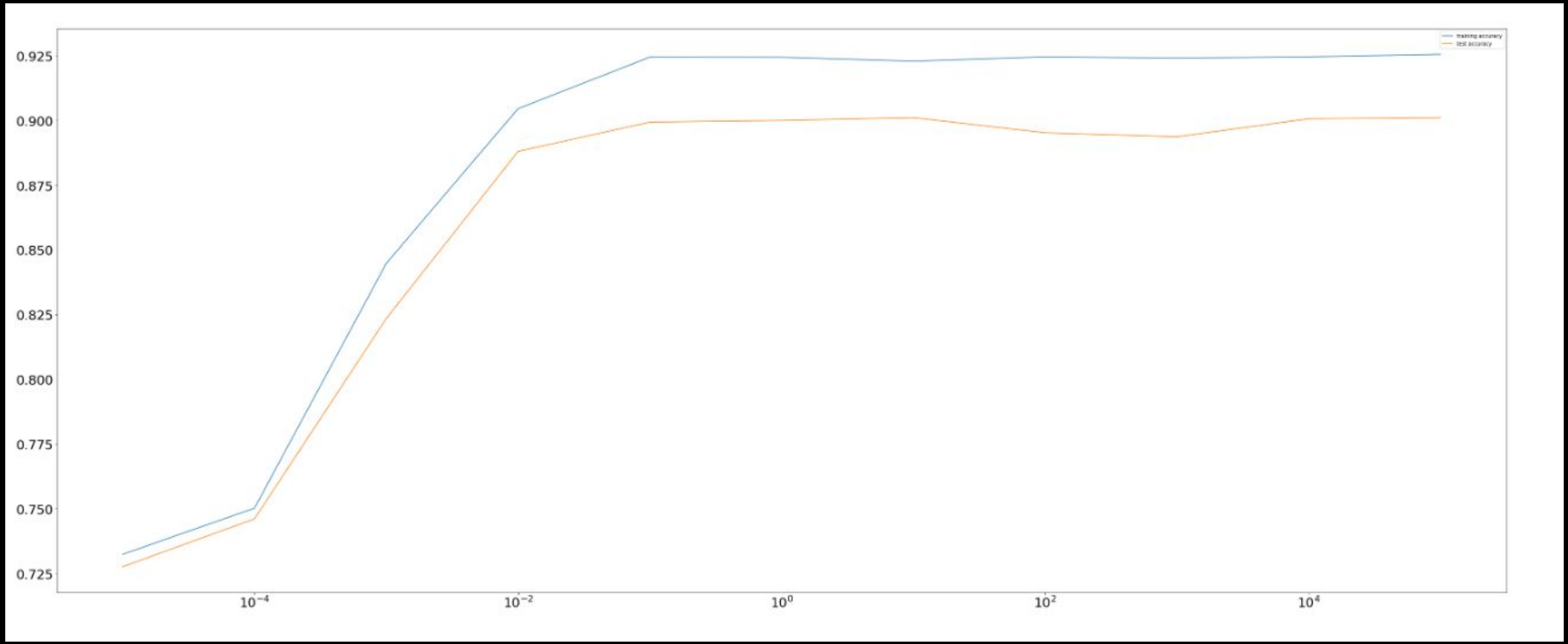


MACHINE LEARNING MODELS

-

LOGISTIC REGRESSION MODEL : GRIDSEARCH

Validation Curve



Accuracy :
0.9255434782608696

LOGISTIC REGRESSION MODEL: RECURSIVE FEATURE ELIMINATION (RFE)

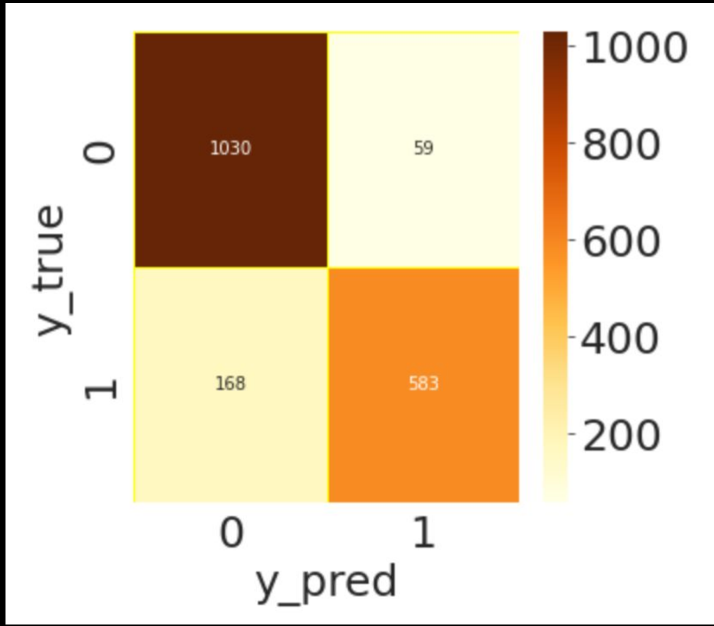
Model

Feature selection approach fits a model and eliminates the weakest feature/features until the desired amount of features is attained.

Observation : a small portion of the features are relevant to our model.

Accuracy: 0.8766304347826087

Confusion Matrix



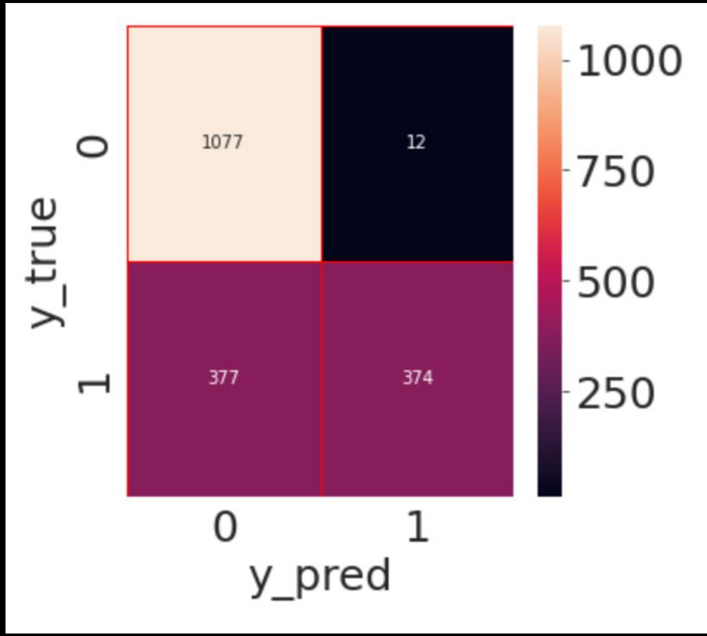
K-NEAREST NEIGHBOR :

Model

Assigns a value to the latest data point based on how closely it resembles the points in the training set.

Accuracy : 0.8130434782608695

Confusion Matrix



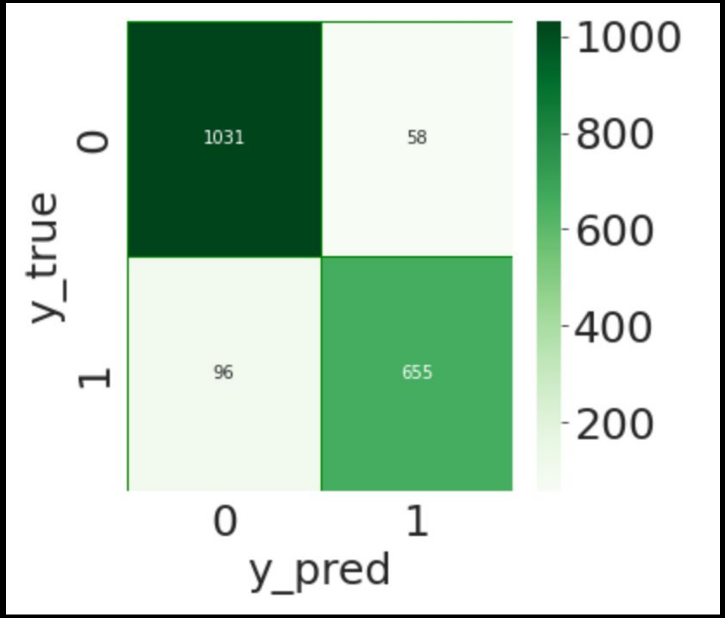
DECISION TREE

Model

Piecewise constant approximation.

Accuracy : 0.9163043478260869

Confusion Matrix



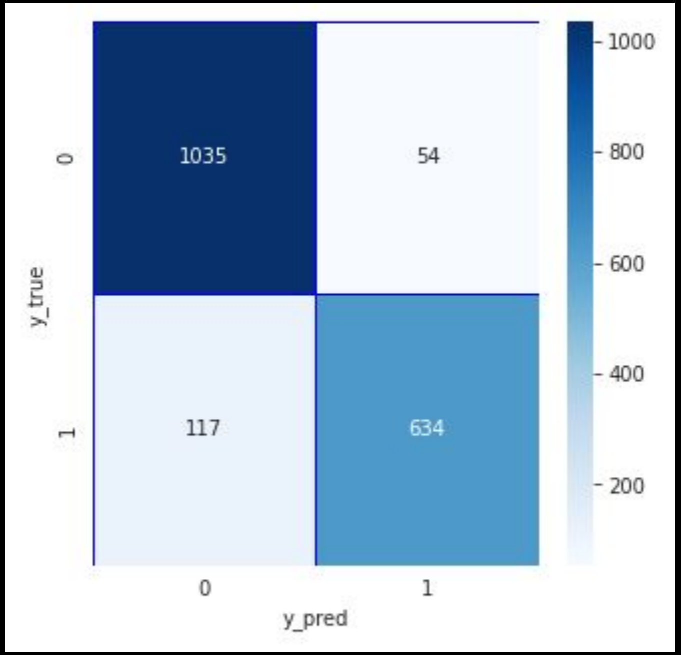
GRADIENT BOOSTING CLASSIFICATION:

Model

Produces a prediction model in the form of an ensemble of weak prediction models.

```
Learning rate:  1
Accuracy score (training): 0.926
Accuracy score (validation): 0.917
```

Confusion Matrix



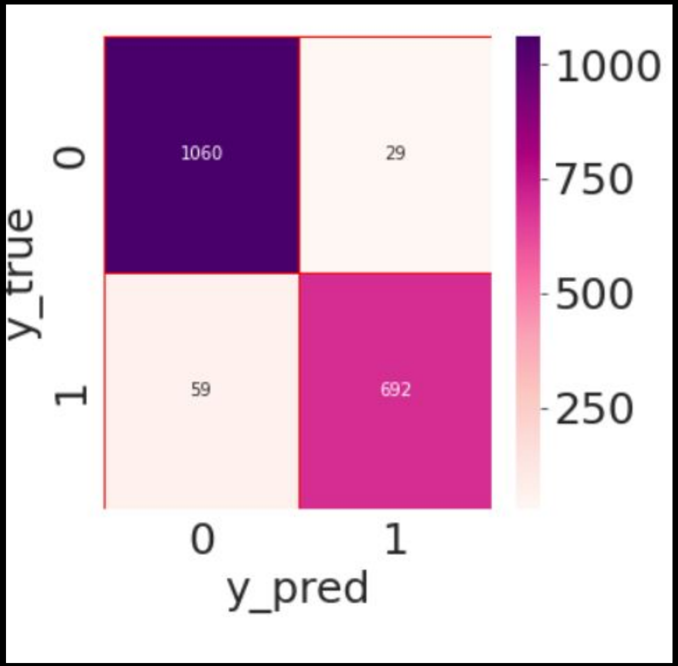
Random Forest

Model

Builds decision trees
Average => Classification
Majority vote => Regression

Accuracy : 0.9521739130434783

Confusion Matrix

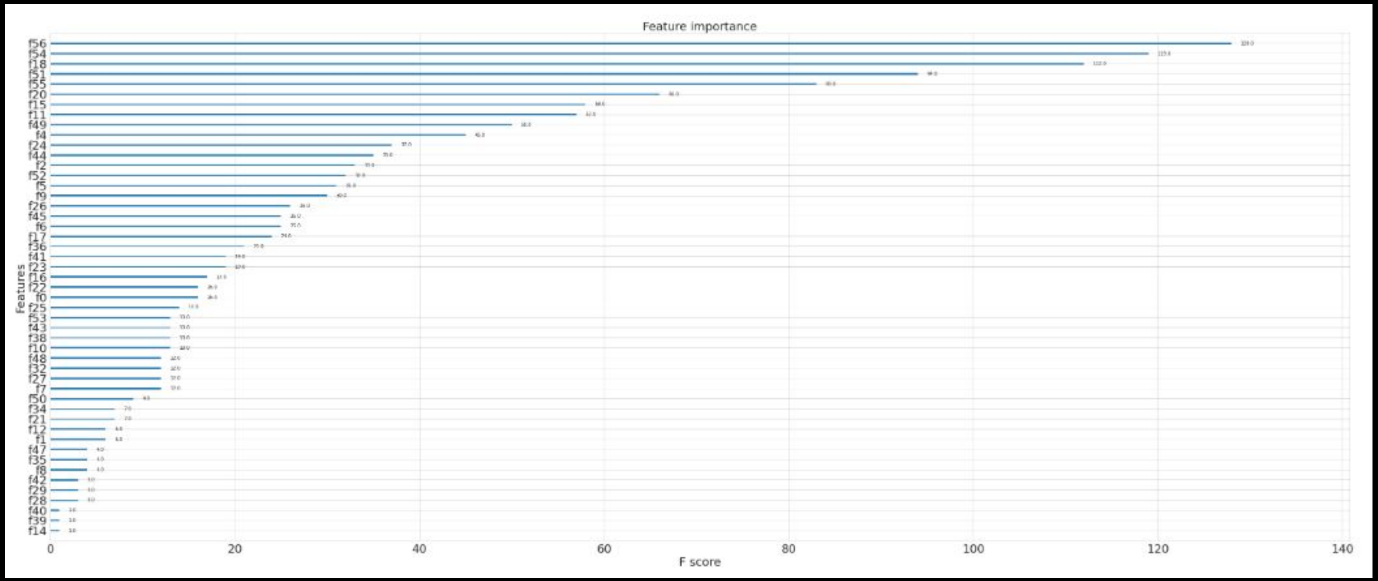


XGBOOST

low false positive and false negative
accuracy of ~95%

Model

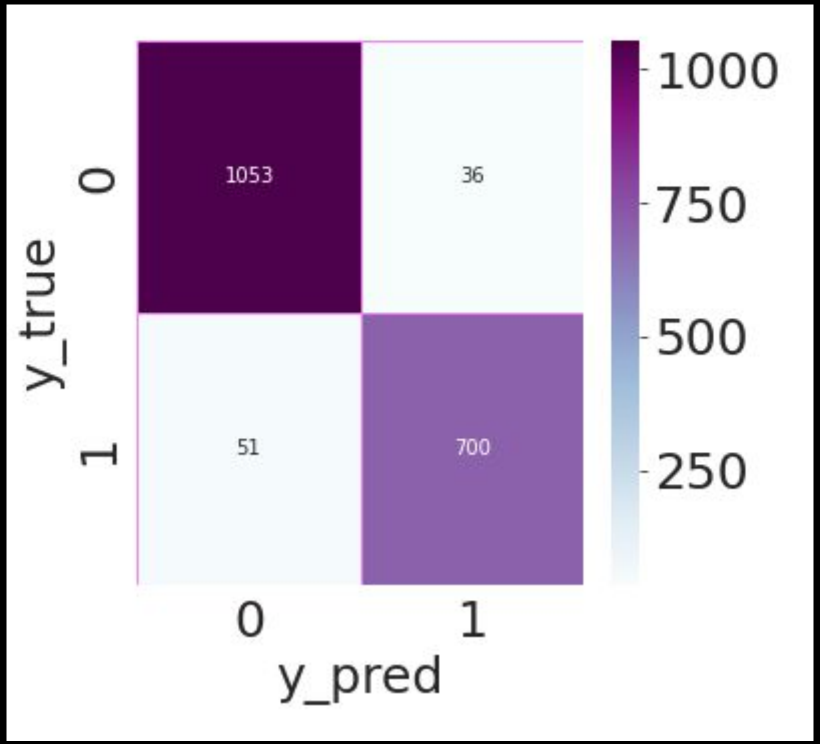
XGBoost is an open-source software library that implements optimized distributed gradient boosting machine learning algorithms under the Gradient Boosting framework.

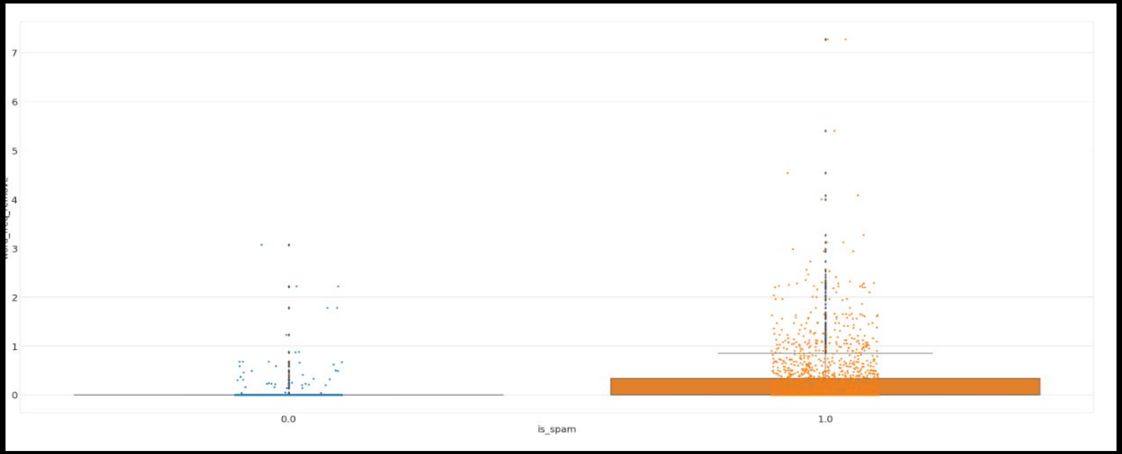


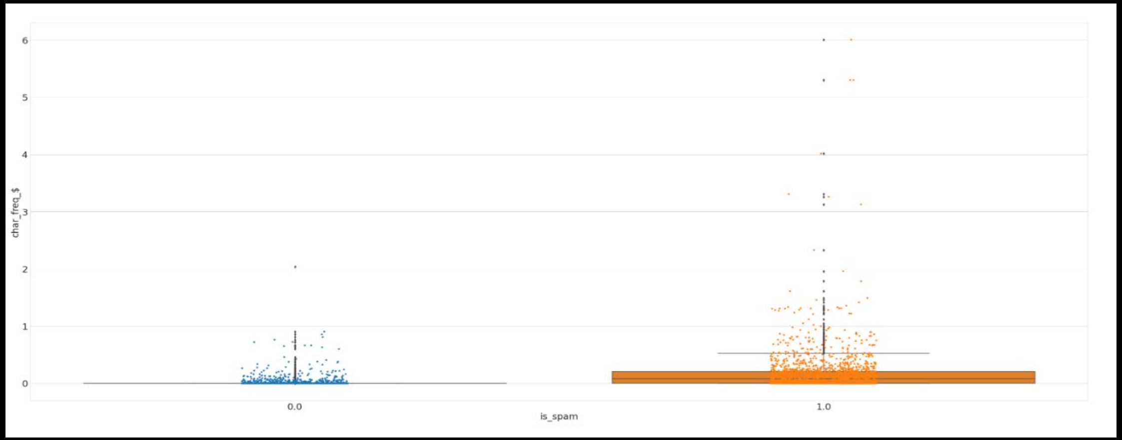
F Score

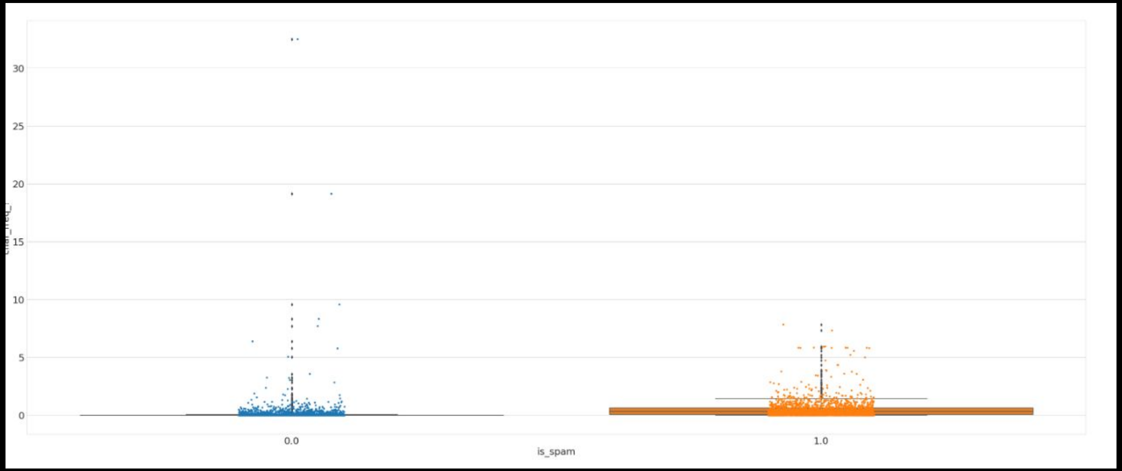
`plot_importance: most important features in the dataset, currently capital_run_length which influences the most the output`

Confusion Matrix









XGBOOST

1. word_freq_remove
2. char_freq_$
3. char_freq_!

Boxplotting the most impacting features.

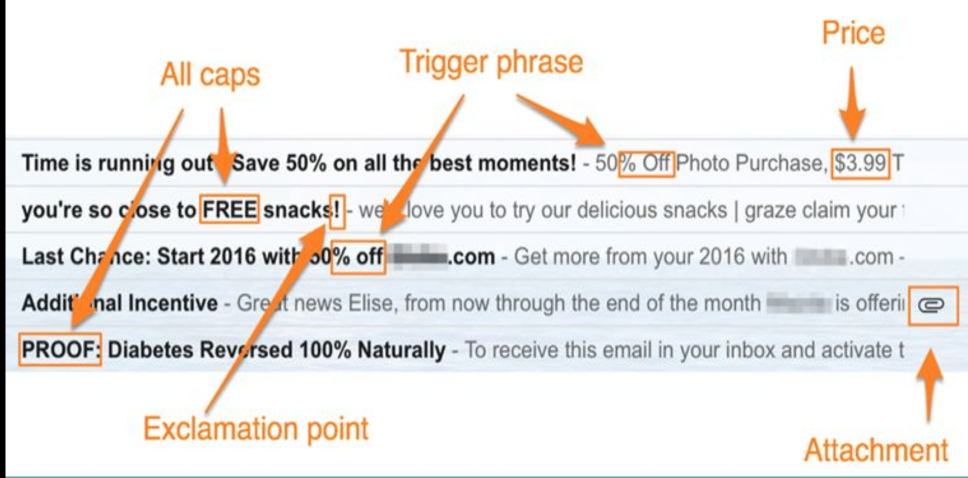
How did we improve our models ?

We :

1. Optimized the hyperparameters

2. Limited our use to impactful data

3. Prioritized cross-validation methods



So What ?

Our Answer

How can we recognize spam ?

Recurrent features, here being :

- 'Remove' : virus scanning apps / virus removal scams
- '$' : fraudulent activities perpetrated by hackers, spread by spam
- '!' : symbolizes excitement, urgency in marketing

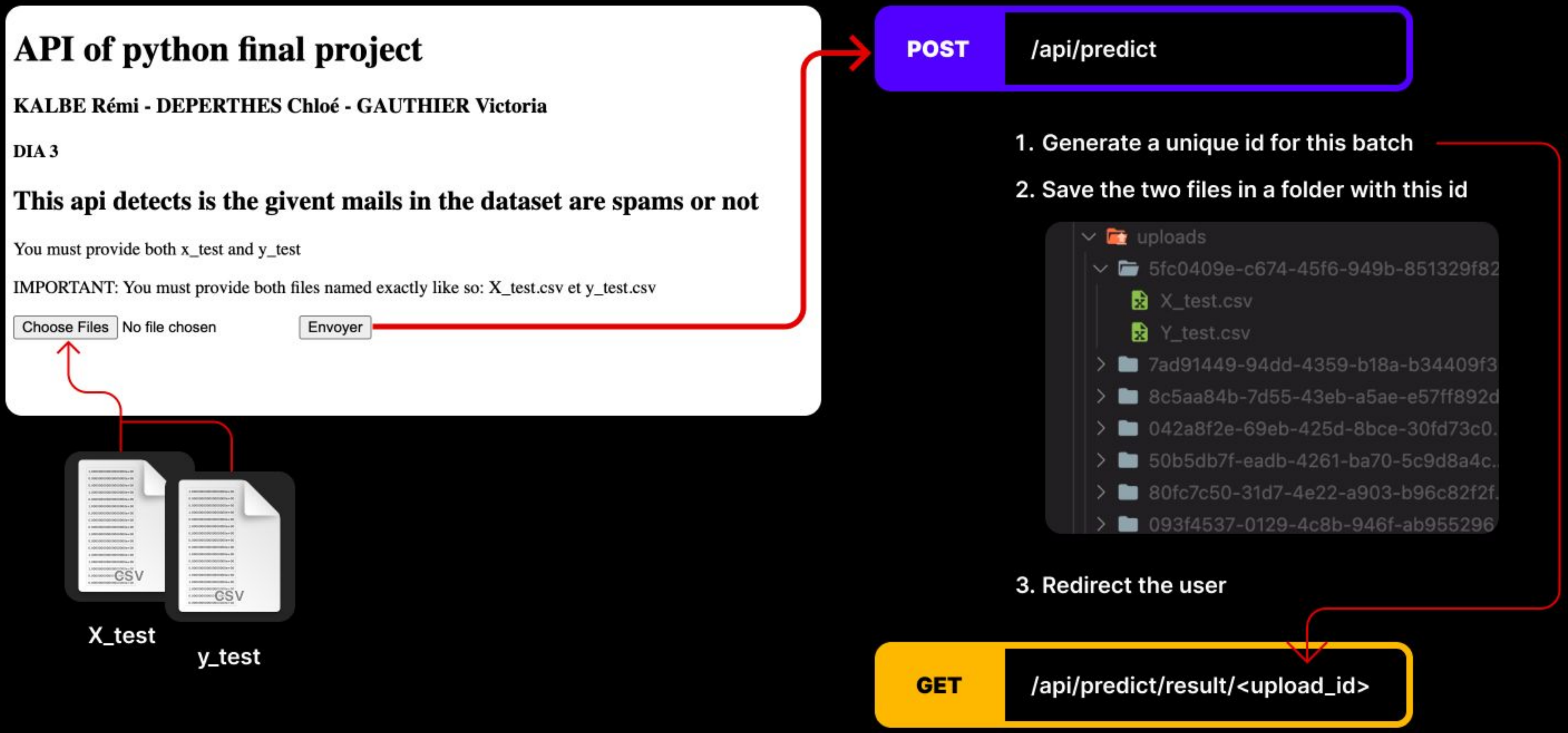
What is the best method to do so ?

MODEL ACCURACY COMPARISON :

	3
random forest	0.9521739130434783
XGBoost	0.95
logistic regression with gridsearch	0.9255434782608696
decision tree	0.9179347826086957
gradient boosting	0.917
Logistic Regression RFE	0.9163043478260869
KNN	0.8010869565217391

Random Forest + XGBoost

API


API of python final project
KALBE Rémi - DEPERTHES Chloé - GAUTHIER Victoria
DIA 3
This api detects is the givent mails in the dataset are spams or not
You must provide both x_test and y_test
IMPORTANT: You must provide both files named exactly like so: X_test.csv et y_test.csv
Choose Files
No file chosen
Envoyer
CSV
CSV
X_test
y_test
POST
/api/predict
1. Generate a unique id for this batch
2. Save the two files in a folder with this id
uploads
5fc0409e-c674-45f6-949b-851329f82
X_test.csv
Y_test.csv
7ad91449-94dd-4359-b18a-b34409f3
8c5aa84b-7d55-43eb-a5ae-e57ff892d
042a8f2e-69eb-425d-8bce-30fd73c0.
50b5db7f-eadb-4261-ba70-5c9d8a4c.
80fc7c50-31d7-4e22-a903-b96c82f2f.
093f4537-0129-4c8b-946f-ab955296
3. Redirect the user
GET
/api/predict/result/<upload_id>

API

GET /api/predict/result/<upload_id>

1. Get the files using the upload id

```
uploads
  5fc0409e-c674-45f6-949b-851329f82
    X_test.csv
    Y_test.csv
  7ad91449-94dd-4359-b18a-b34409f3
  8c5aa84b-7d55-43eb-a5ae-e57ff892d
  042a8f2e-69eb-425d-8bce-30fd73c0.
  50b5db7f-eadb-4261-ba70-5c9d8a4c.
  80fc7c50-31d7-4e22-a903-b96c82f2f.
  093f4537-0129-4c8b-946f-ab955296
```

2. Get the training files

3. Initialize each of our models

```python
# Initialize the models
model_logreg_rfe = LogisticRegressionModel(
    X_train, y_train, X_test, y_test, LRMUsing.RFE)
model_logreg_grid = LogisticRegressionModel(
    X_train, y_train, X_test, y_test, LRMUsing.GridSearchCV)
model_knn = KNNModel(X_train, y_train, X_test, y_test)
model_decision_tree = DecisionTreeModel(X_train, y_train, X_test, y_test)
model_xg_boost = XGBoostModel(X_train, y_train, X_test, y_test)
model_gradient_boosting = GradientBoostingModel(
    X_train, y_train, X_test, y_test)
```

```python
def __init__(self, X_train, y_train, X_test, y_test):
    self.X_train = X_train
    self.y_train = y_train
    self.X_test = X_test
    self.y_test = y_test

    # convert y_train and y_test to int
    self.y_train = self.y_train.astype(int)
    self.y_test = self.y_test.astype(int)

    self.model = neighbors.KNeighborsRegressor(n_neighbors=5)
    self.model = self.model.fit(self.X_train, self.y_train)
```

4. Get their score

5. Get their prediction and save it in a file

- We create a result folder with the same id as the upload id
- We write a csv file for each model's predictions

```
results
  5fc0409e-c674-45f6-949b-851329f82...
    decision_tree_predictions.csv
    gradient_boosting_predictions.csv
    knn_predictions.csv
    logreg_grid_predictions.csv
    logreg_rfe_predictions.csv
    xg_boost_predictions.csv
  042a8f2e-69eb-425d-8bce-30fd73c0...
  093f4537-0129-4c8b-946f-ab955296...
  404e599f-8ce3-4921-8e57-b303b6dd...
```

API

GET /api/predict/result/<upload_id>

6. Get their accuracy

7. Get their confusion matrix, then we plot it and save it as a base64 image

```
# Get the confusion matrix
model_logreg_rfe_confusion_matrix = model_logreg_rfe.confusion_matrix_base64()
model_logreg_grid_confusion_matrix = model_logreg_grid.confusion_matrix_base64()
model_knn_confusion_matrix = model_knn.confusion_matrix_base64()
model_decision_tree_confusion_matrix = model_decision_tree.confusion_matrix_base64()
model_xg_boost_confusion_matrix = model_xg_boost.confusion_matrix_base64()
model_gradient_boosting_confusion_matrix = model_gradient_boosting.confusion_matrix_base64()
```

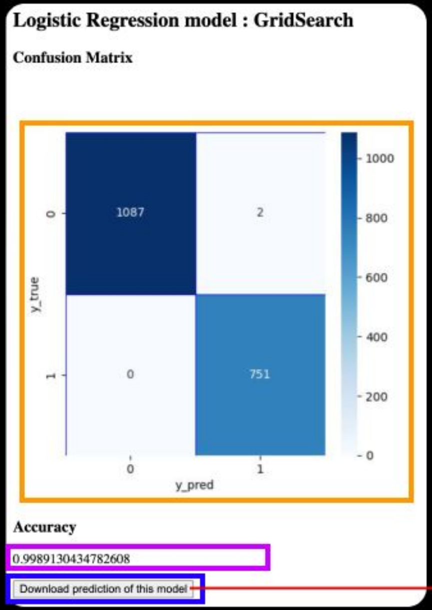
8. Decode each base64 images and sent it to the result page template with all the other variables

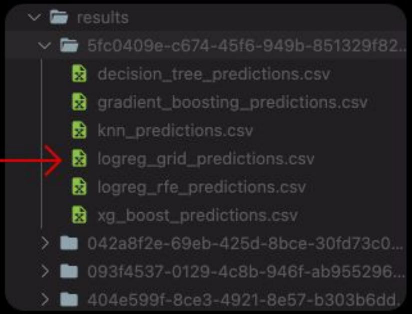
```
return render_template('results.html',
                       logreg_rfe_confusion_matrix_base64=model_logreg_rfe_confusion_matrix.decode(
                           'utf8'),
                       logreg_grid_confusion_matrix_base64=model_logreg_grid_confusion_matrix.decode(
                           'utf8'),
                       knn_confusion_matrix_base64=model_knn_confusion_matrix.decode(
                           'utf8'),
                       dt_confusion_matrix_base64=model_decision_tree_confusion_matrix.decode(
                           'utf8'),
                       xgb_confusion_matrix_base64=model_xg_boost_confusion_matrix.decode(
                           'utf8'),
                       gb_confusion_matrix_base64=model_gradient_boosting_confusion_matrix.decode(
                           'utf8'),
                       logreg_grid_accuracy=model_logreg_grid_accuracy,
                       logreg_rfe_accuracy=model_logreg_rfe_accuracy,
                       knn_mae=knn_mae,
                       knn_mse=knn_mse,
                       knn_rmse=knn_rmse,
                       knn_score=model_knn_score,
                       dt_score=model_decision_tree_score,
                       xgb_score=model_xg_boost_score,
                       gb_score=model_gradient_boosting_score,
                       logreg_grid_prediction_url=f'/api/uploads/{upload_id}/logreg_grid',
                       logreg_rfe_prediction_url=f'/api/uploads/{upload_id}/logreg_rfe',
                       knn_prediction_url=f'/api/uploads/{upload_id}/knn',
                       dt_prediction_url=f'/api/uploads/{upload_id}/decision_tree',
                       xgb_prediction_url=f'/api/uploads/{upload_id}/xg_boost',
                       gb_prediction_url=f'/api/uploads/{upload_id}/gradient_boosting',
                       )
```

API

```python
e('results.html',
    logreg_rfe_confusion_matrix_base64=model_logreg_rfe_confusion_matrix.decode(
        'utf8'),
    logreg_grid_confusion_matrix_base64=model_logreg_grid_confusion_matrix.decode
        'utf8'),
    knn_confusion_matrix_base64=model_knn_confusion_matrix.decode(
        'utf8'),
    dt_confusion_matrix_base64=model_decision_tree_confusion_matrix.decode(
        'utf8'),
    xgb_confusion_matrix_base64=model_xg_boost_confusion_matrix.decode(
        'utf8'),
    gb_confusion_matrix_base64=model_gradient_boosting_confusion_matrix.decode(
        'utf8'),
    logreg_grid_accuracy=model_logreg_grid_accuracy,
    logreg_rfe_accuracy=model_logreg_rfe_accuracy,
    knn_mae=knn_mae,
    knn_mse=knn_mse,
    knn_rmse=knn_rmse,
    knn_score=model_knn_score,
    dt_score=model_decision_tree_score,
    xgb_score=model_xg_boost_score,
    gb_score=model_gradient_boosting_score,
    logreg_grid_prediction_url=f'/api/uploads/{upload_id}/logreg_grid',
    logreg_rfe_prediction_url=f'/api/uploads/{upload_id}/logreg_rfe',
    knn_prediction_url=f'/api/uploads/{upload_id}/knn',
    dt_prediction_url=f'/api/uploads/{upload_id}/decision_tree',
    xgb_prediction_url=f'/api/uploads/{upload_id}/xg_boost',
    gb_prediction_url=f'/api/uploads/{upload_id}/gradient_boosting',
)
```

```html
<div>
  <h2>Logistic Regression model : GridSearch</h2>
  <h3>Confusion Matrix</h3>
  <img src="data:image/png;base64,{{logreg_grid_confusion_matrix_base64}}" alt="">
  <h3>Accuracy</h3>
  <p>{{logreg_grid_accuracy}}</p>
  <a href="{{logreg_grid_prediction_url}}" download>
    <button href="{{logreg_grid_prediction_url}}" download="logreg_grid_predictions.csv">Download prediction of this model</button>
  </a>
  </div>
<div>
```





Spam, phishing, arnaques : signaler pour agir

Signal spam + tentatives d’escroqueries : https://www.internet-signalement.gouv.fr/PharosS1/etape/contenu

THANK YOU !

Chloé DEPERTHES, Victoria GAUTHIER, Rémi KALBE