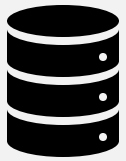
SPAMBASE DATA SET

Syrine Bouchelleghem

Hajar Azzouzi



THE DATASET

https://archive.ics.uci.edu/ml/datasets/Spambase

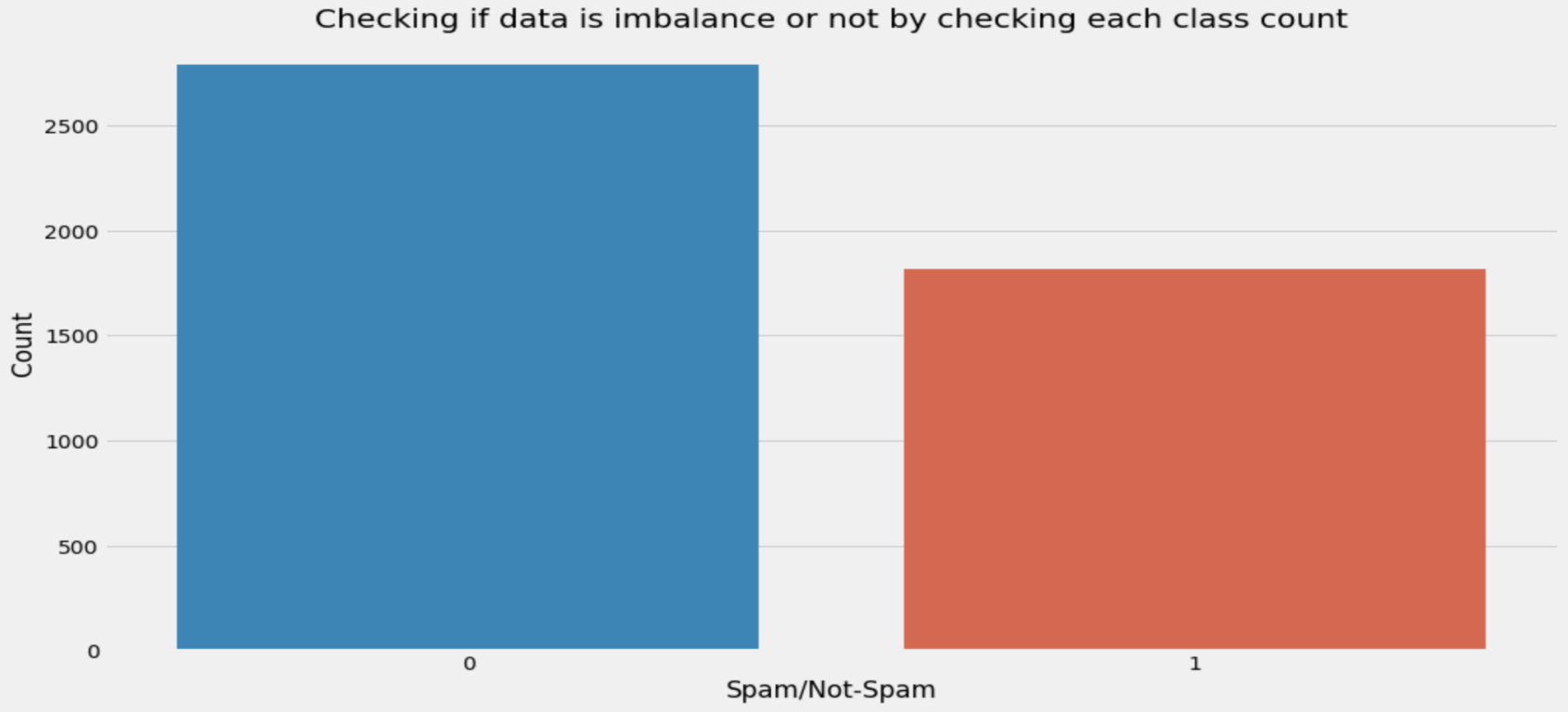
- Classification : Spam (1) / No Spam (0)
- 57 float attributes
- 1 nominal {0,1} class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0)

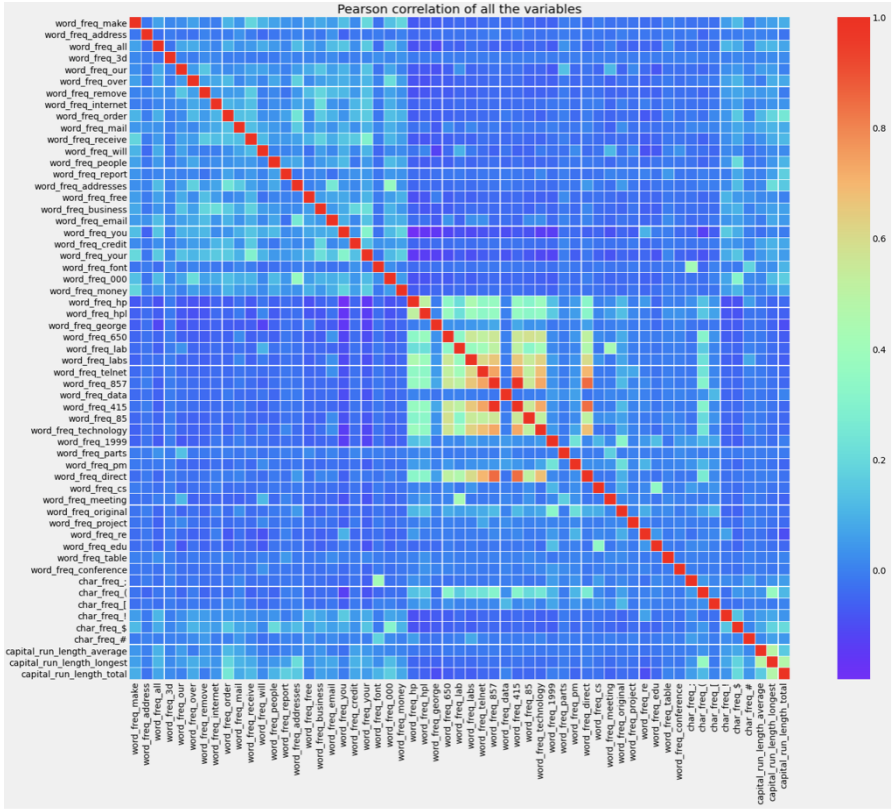
The goal of this project is to :

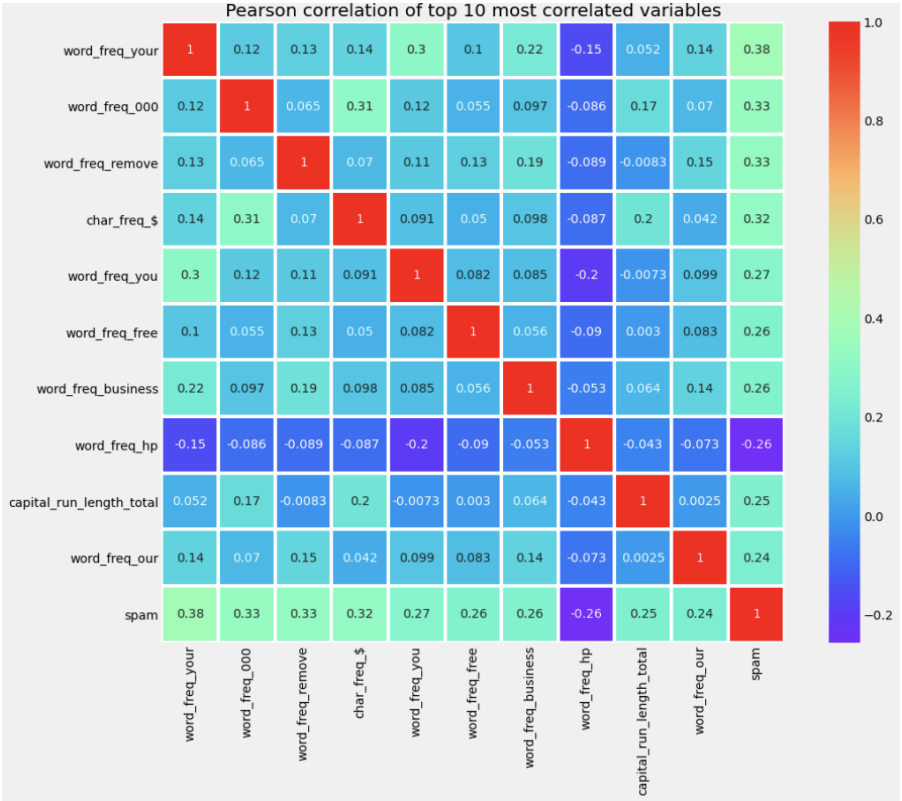
- visualize the link between the variables and the target by using different libraries.
- Testing Machine Learning models on the data set to predict the class of an email (use the scikit-learn library)
- Transform the model into a Django API

THE DATASET

https://archive.ics.uci.edu/ml/datasets/Spambase



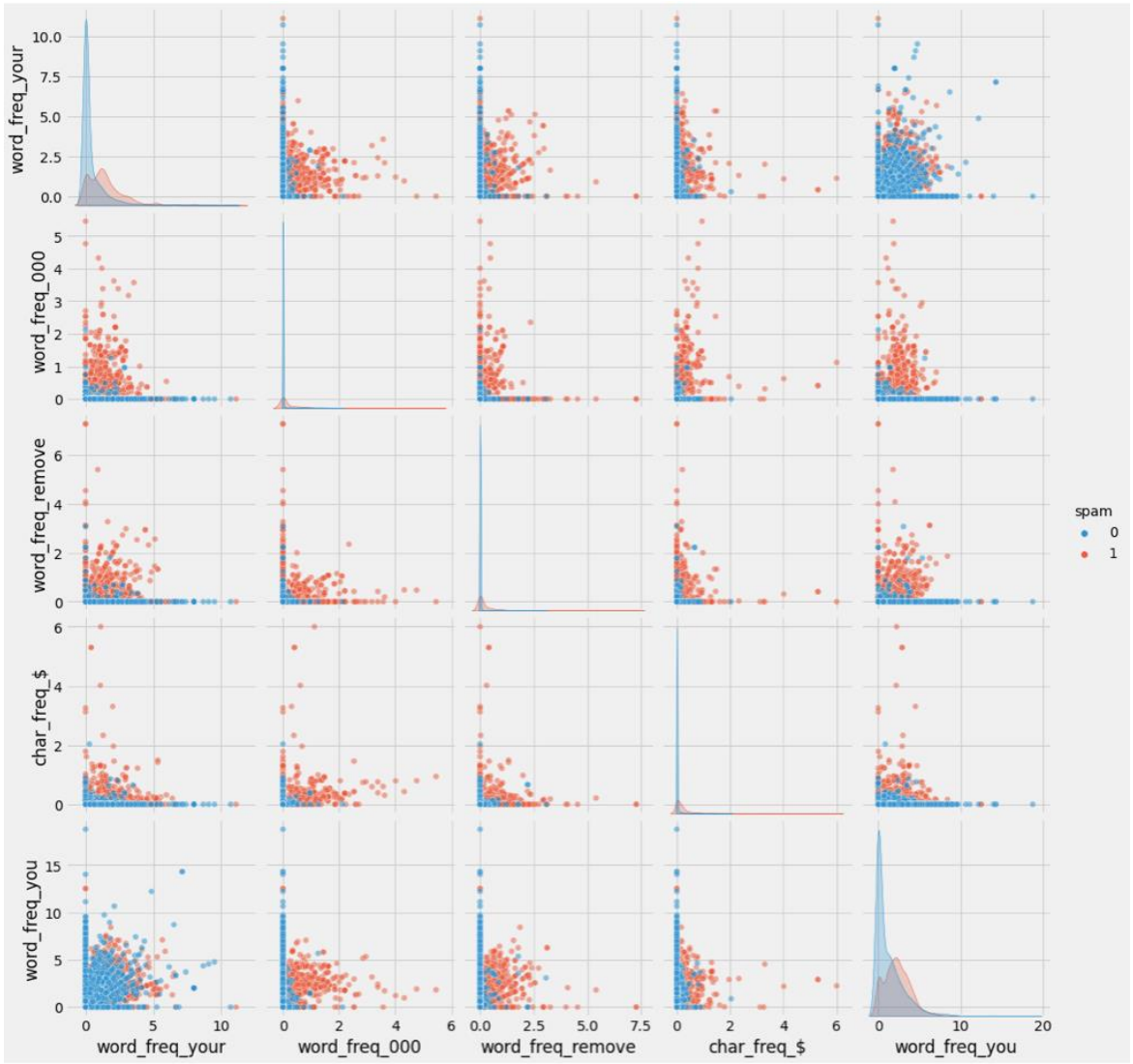




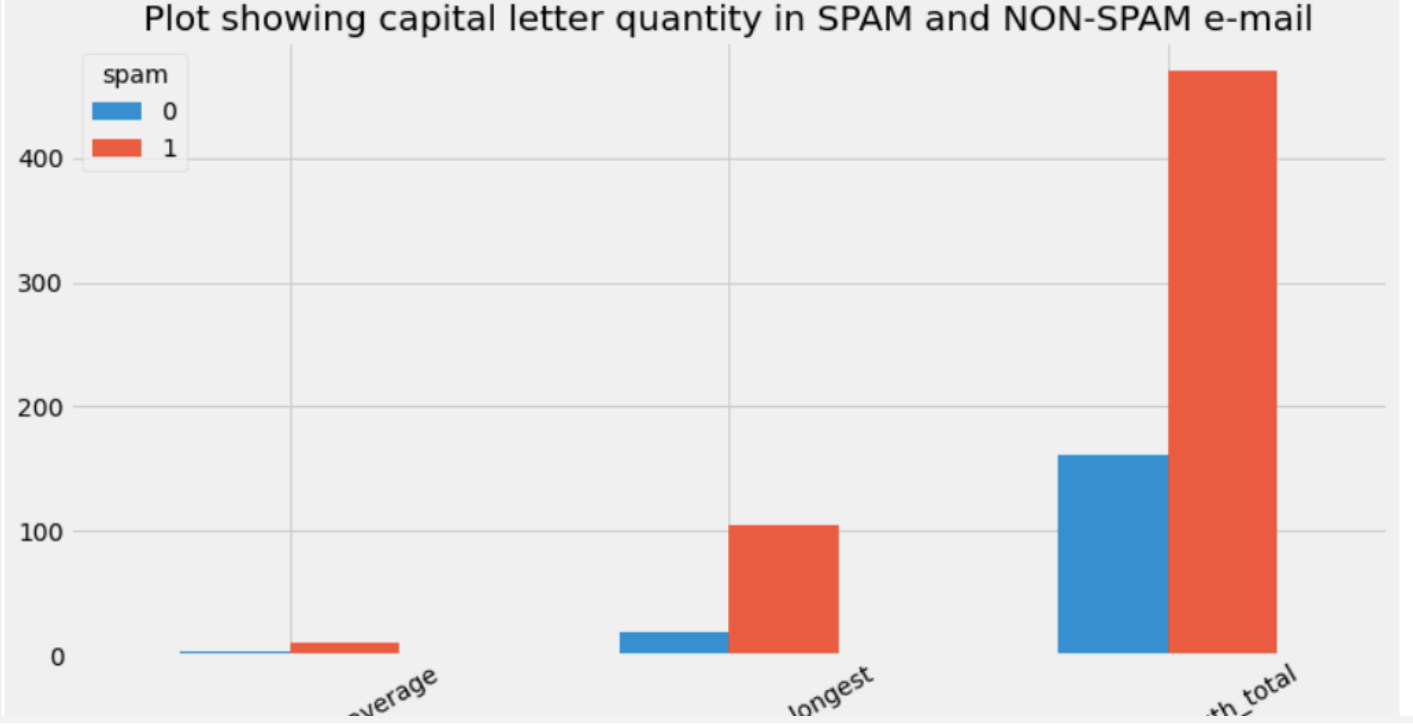
VARIABLES CORRELATION

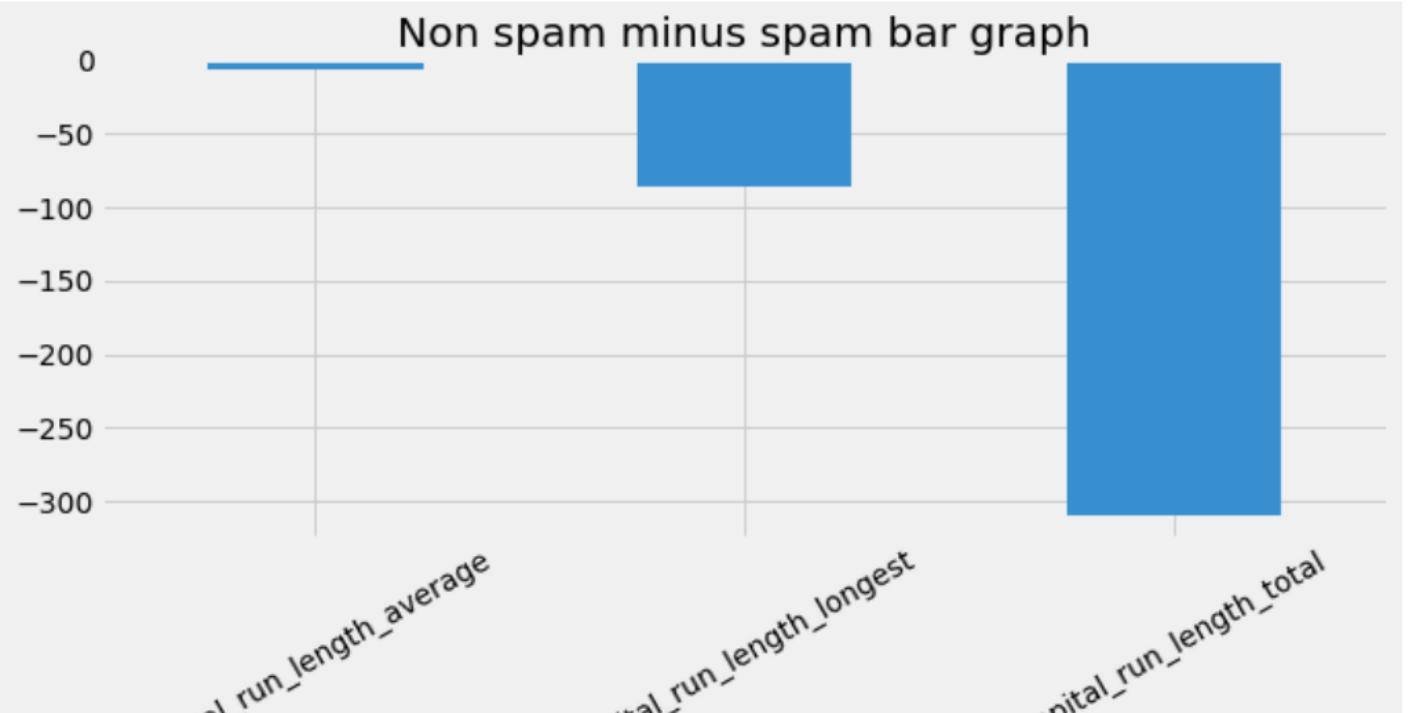
We can notice that a lot of variables not correlated to the target. Not using them when fitting the model can improve performance.

PLOT OF TOP FIVE MOST CORRELATED VARIABLES WITH DEPENDENT VARIABLE



SPAM VS
NON
SPAM





MODELING

First, we tested classification models **without processing any data** of our dataset.

The following models were tested :

- Random Forest
- Quadratic Discriminant Analysis
- K-Nearest Neighbors*
- Boosting
- Bagging
- Logistic Regression
- Linear Discriminant Analysis

	Unprocessed
Random Forest	0.947871
Quadratic discriminant Analysis	0.793223
KNN (k=1)	NaN
Boosting	0.915725
Bagging	0.93397
Logistic regression	0.903562
Linear Discriminant Analysis	0.873154
KNN (k=7)	0.81755

The score used to compare models is the accuracy.

With raw data, RadomForest gave the best predictions.

* the KNN classifier was tested with k from 1 to 25 and the value with best score only is kept

Then, we tested classification models with **processed data**.

The training data is scaled.

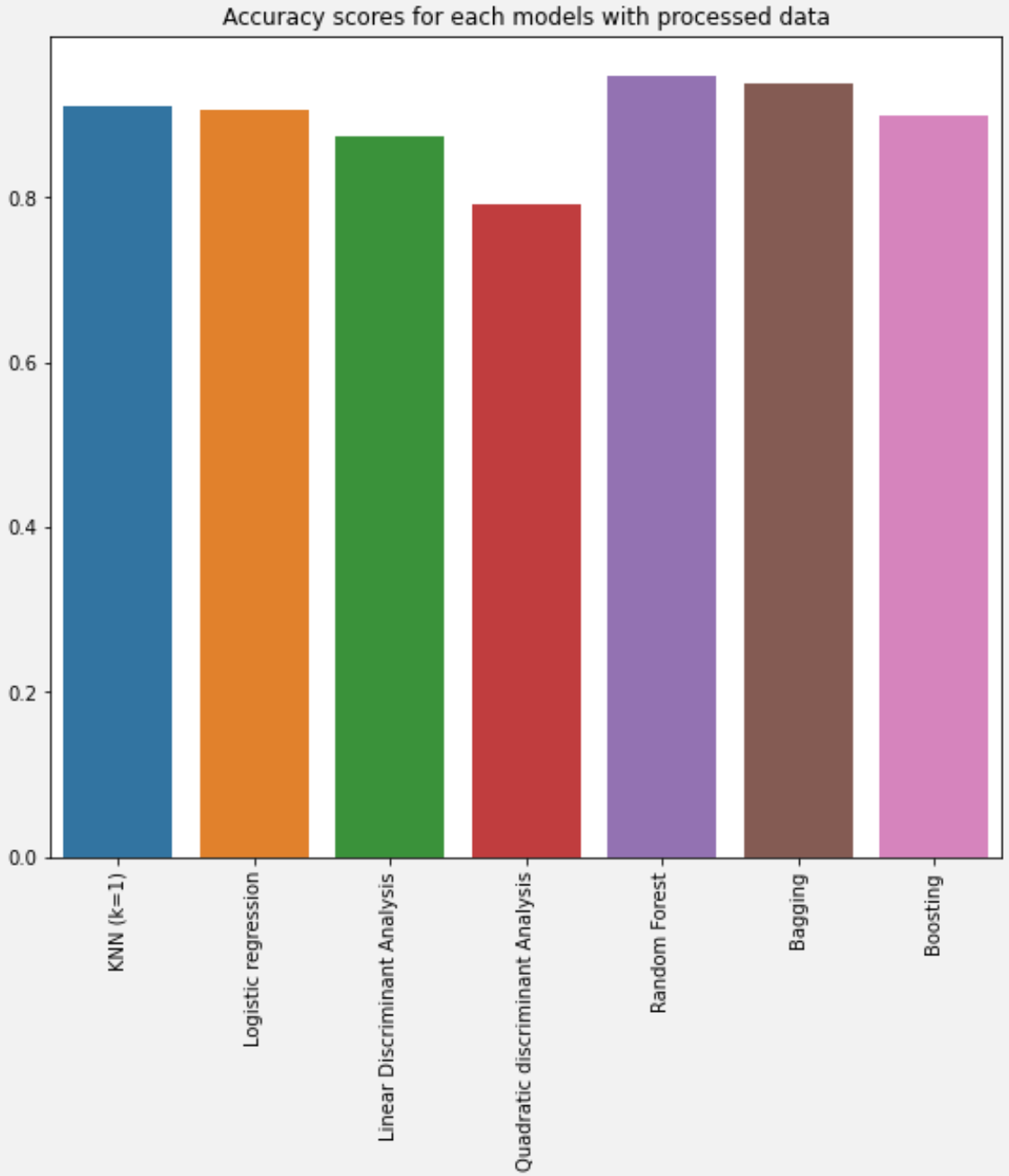
The following models were tested :

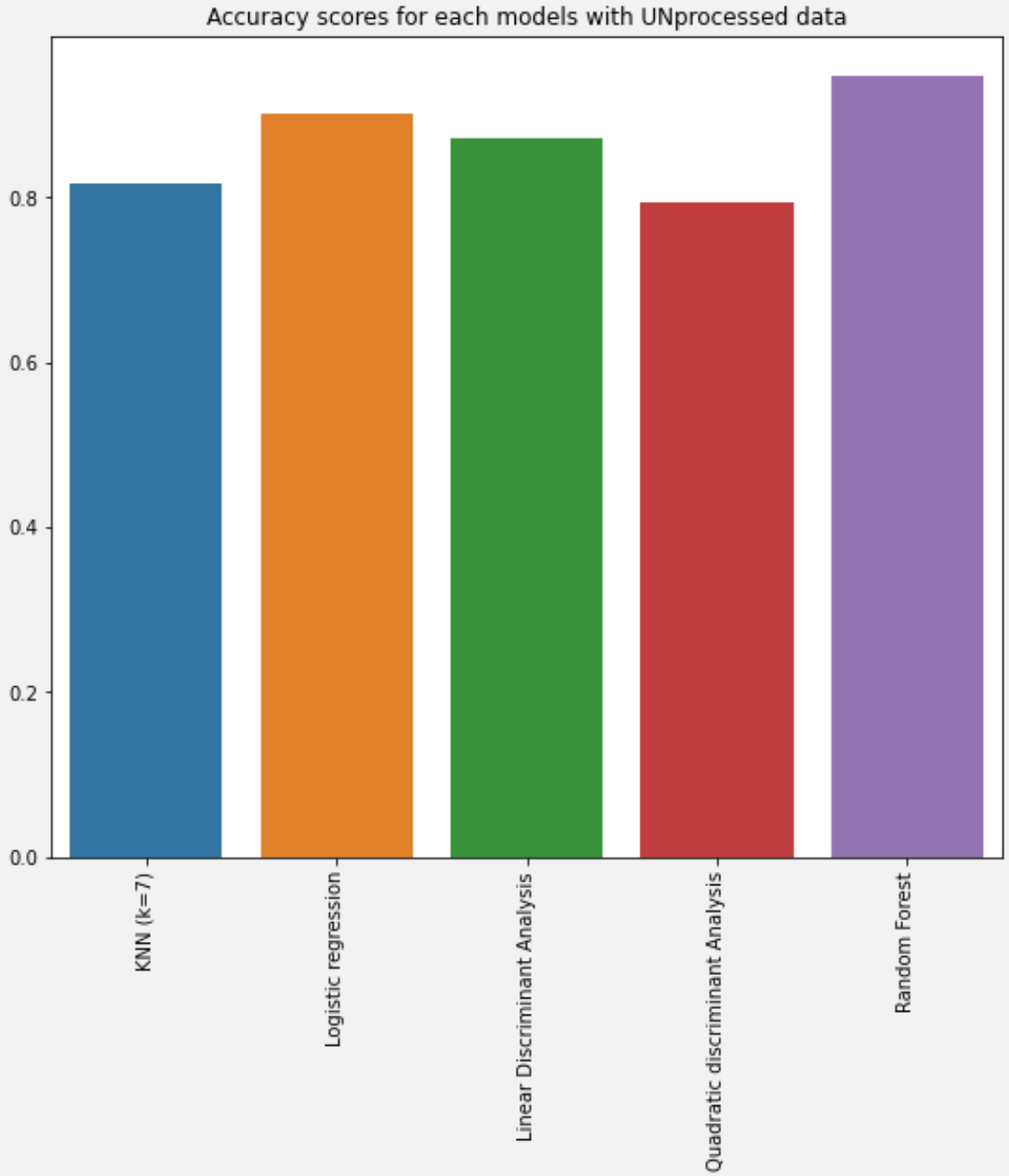
- Random Forest
- Quadratic Discriminant Analysis
- K-Nearest Neighbors*
- Boosting
- Bagging
- Logistic Regression
- Linear Discriminant Analysis

	Unprocessed	Processed
Random Forest	0.947871	0.947003
Quadratic discriminant Analysis	0.793223	0.791486
KNN (k=1)	NaN	0.910513
Boosting	0.915725	0.898349
Bagging	0.93397	0.945265
Logistic regression	0.903562	0.904431
Linear Discriminant Analysis	0.873154	0.873154
KNN (k=7)	0.81755	NaN

With processed data, RadomForest also gave the best predictions.

* the KNN classifier was tested with k from 1 to 25 and the value with best score only is kept


Accuracy scores for each models with processed data
0.0
0.2
0.4
0.6
0.8
KNN (k=1)
Logistic regression
Linear Discriminant Analysis
Quadratic discriminant Analysis
Random Forest
Bagging
Boosting


Accuracy scores for each models with UNprocessed data
0.0
0.2
0.4
0.6
0.8
KNN (k=7)
Logistic regression
Linear Discriminant Analysis
Quadratic discriminant Analysis
Random Forest

CONCLUSION

Finally, the best scores were obtained with the RandomForest model that was trained with processed data.

RandomForest model can also be improved by changing the number of trees (n_estimators).

We tested the model with a number of trees from 100 to 1000 and got the number that gives the best performance.

```
scores ={}
for k in range(100,1000):
        rf = RandomForestClassifier(n_estimators=k)
        rf.fit(x_train,y_train.values.ravel())
        scores[k]=rf.score(x_test,y_test)

print((max(scores, key=scores.get),max(scores.values())))
```

Final best score :0.950477845351868
(RandomForest with 111 trees)