

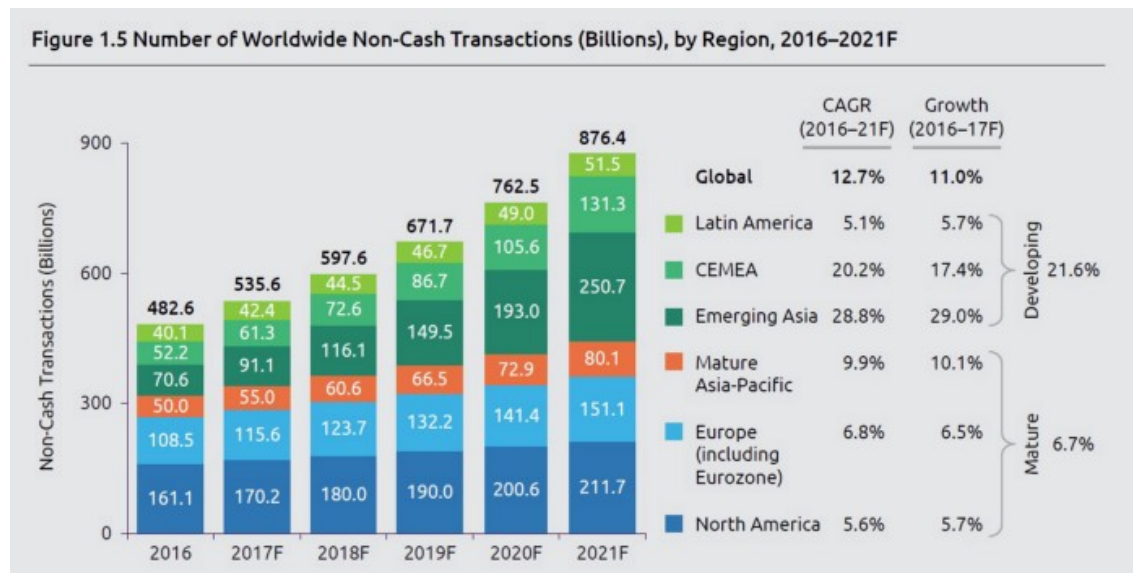
# Credit Card Fraud Detection (Project Report)

List of items:

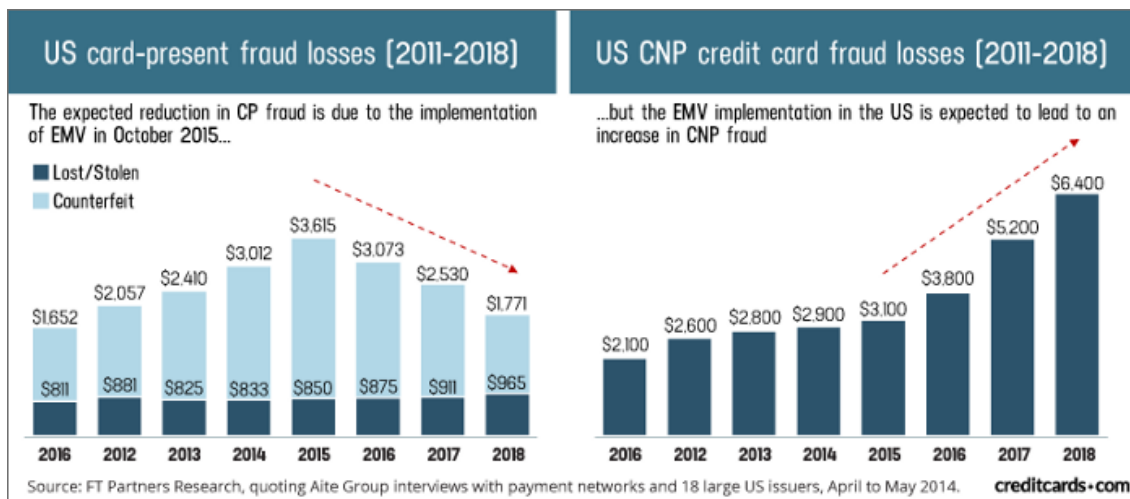
- Introduction to CCFD
- Model Training
- Web App Creation
- Conclusion
- References

## Introduction

In today's world, we are on the express train to a cashless society. According to the World Payments Report, in 2016 total non-cash transactions increased by 10.1% from 2015 for a total of 482.6 billion transactions! That's huge! Also, it's expected that in future years there will be a steady growth of non-cash transactions as shown below:



Now, while this might be exciting news, on the flip-side fraudulent transactions are on the rise as well. Even with EMV smart chips being implemented, we still have a very high amount of money lost from credit card fraud:



This is now becoming a serious problem since most of the time, a person who has become a victim of this fraud don't have any idea about what has happened until the very end.

So in this project, what we have tried is to create a Web App for the detection of such type of frauds with the help of Machine Learning.

In the following sections, we will be explaining about the creation and importance of both a good Machine Learning model and the Web App.

## Model Training

### Understanding the data and related constraints

Since the data for this project is very unbalanced due to the fact that number of cases of Fraud transactions are very low in comparison to number of cases of Valid transactions makes the model training a bit hectic. Because if we consider "classification accuracy" as the metric for training, we won't be getting the perfect view of how much our model is learning, because the classification accuracy is derived as:

$$\text{classification\_accuracy} = \frac{\text{No. of correct predictions}}{\text{No. of labels}}$$

So, now, consider the fact that if our data have 98% of the values to be valid while only 2% to be frauds, if our model predicts all values to be valid, it will eventually achieve 98% accuracy at the end of the day, but the model will be an absolute wastage.

For this reason, we use a different type of metric which will give us much more important information about what our model has learned.

Actually, what we do is we print the classification matrix for our model predictions and then we judge our model based on that matrix.

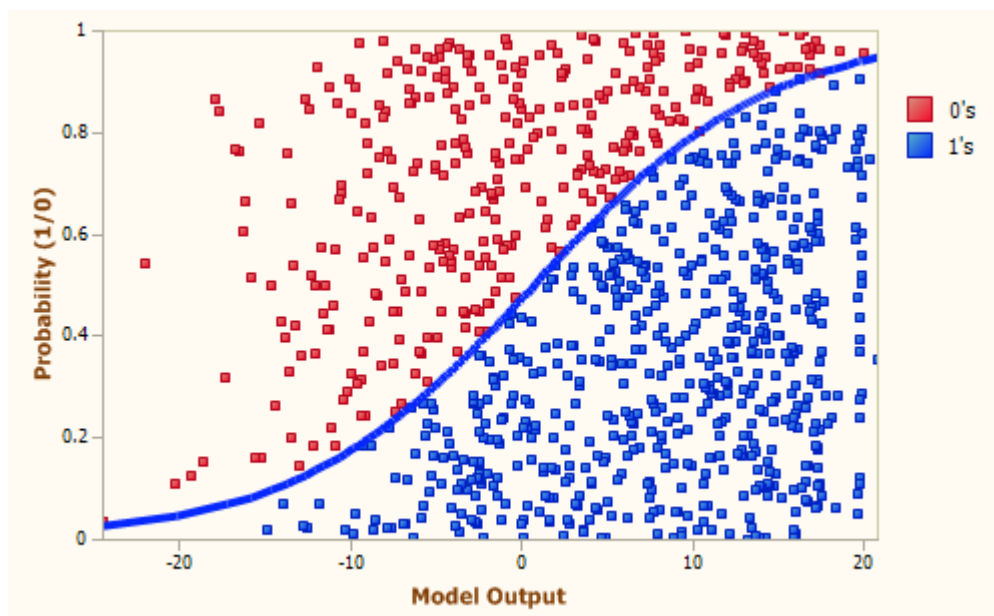
Precision and Recall are two of the derivatives of a confusion matrix, we will consider them both though, but only Recall will come in handy for us since high Recall will ensure that no fraud value gets detected to be a valid one. Also, Precision do the vice-versa. We will need to find the best threshold where the Precision-Recall tradeoff will give us the best results.

## Preprocessing data

- **Balancing data** Since the data is very imbalanced, we will be using some Undersampling and Oversampling techniques. As the name suggests, Undersampling is use to reduce the samples from majority class and Oversampling is used to increase the samples from minority class. This way, we can achieve some balancing of the data.
- **Scaling features** Now, even though almost all of the features are dimensionally reduced using some dimensionality reduction technique, two of the features are in their original form. Time and Amount are the two features which we will be scaling in order to make our model learn features correctly.
- **Splitting the data** Now, since we needed to save some entries from the data for our testing purpose, we will now be splitting the data into two parts, namely, `train` and `test`.
- **Miscellaneous** Now, other than the above three techniques, we did some Exploratory Data Analysis[EDA] on the data, we get the idea of outliers in the data, feature importance etc. by doing this amazing part. One can find the EDA in the notebook itself.

## Model Architecture

We will be using a Logistic Regression classifier for our project. A logistic Regression model is used to predict the probability of a certain class or event existing. We then decide the class from which the entry belongs by using a threshold value. This threshold value is decided by manipulating the Precision-Recall tradeoff as explained above.

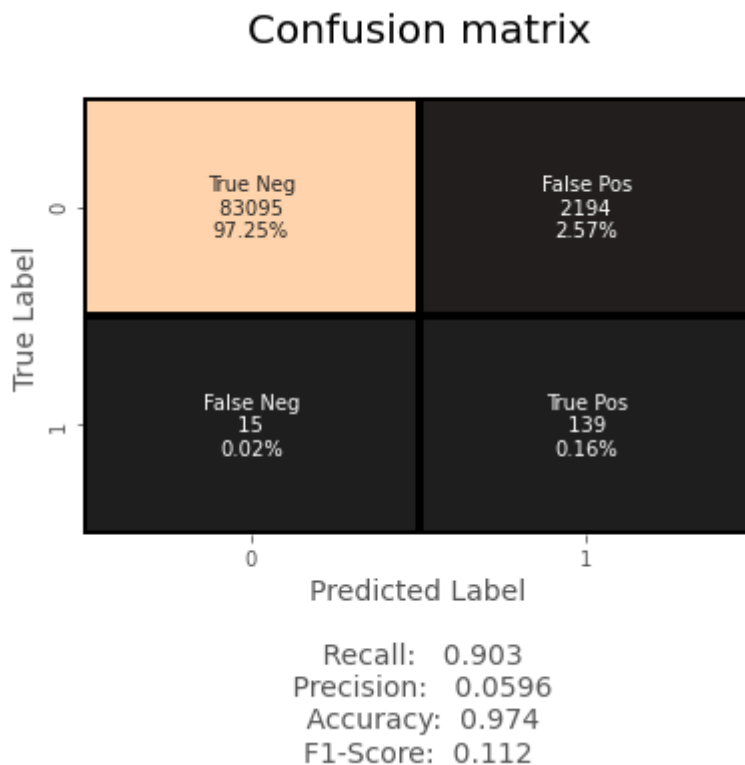


Also, while training, we used GridSearchCV to find the best possible parameters for our model so that it can squeeze the maximum amount of important information out of the data.

## Post Training

After a model gets trained, this is our duty to understand the results and ensure its reliability because this will be used for a generalization purpose.

Let's now see the inference results we got after training by using the following confusion matrix from our training notebook:



## Creating Web App

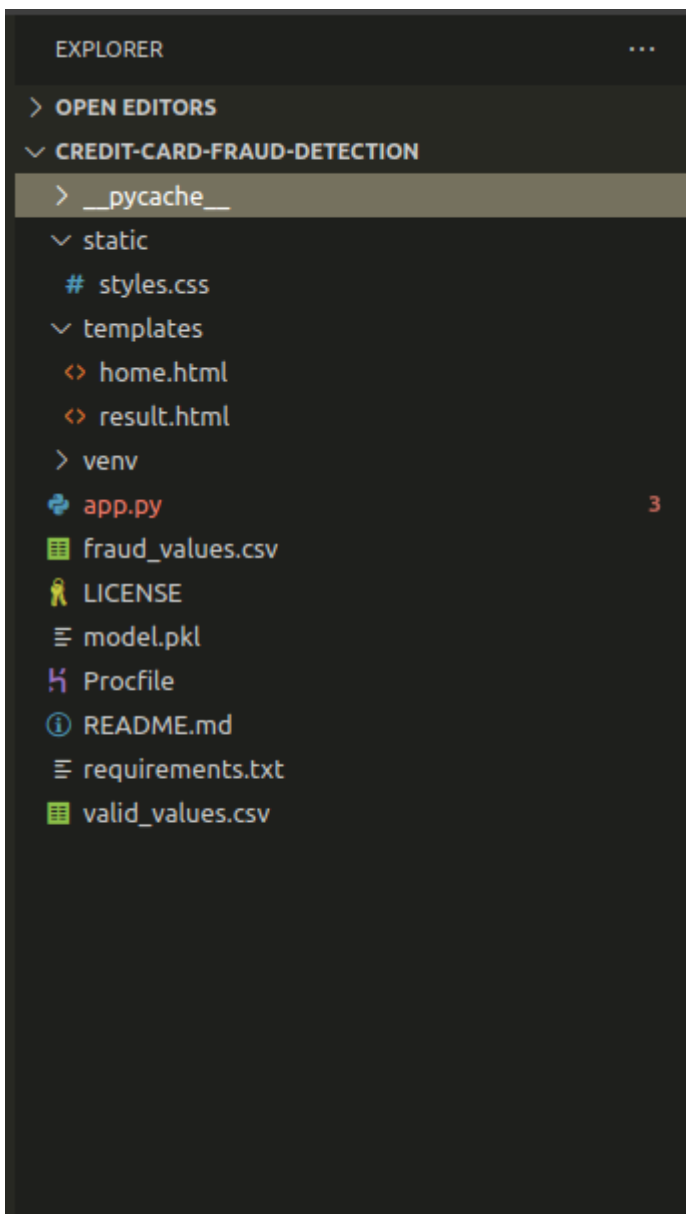
### Brief Introduction to a Web App

A web application is an application software that runs on a web server, unlike computer-based software programs that are stored locally on the Operating System of the device. Web applications are accessed by the user through a web browser with an active internet connection. These applications are programmed using a client-server modeled structure—the user is provided services through an off-site server that is hosted by a third-party. The third party whose services we are using is Heroku, Heroku is a great place to launch your apps upto a limited size. We could not do this project without the support of Heroku who gave us the opportunity to host our web app.

### Behind the scenes

In case of any web app, everyone sees the front page of it, we will discuss that in the next section, in this section, we will be discussing what is happening in behind.

We are using various files to host this web app, the directory structure can be seen as below:

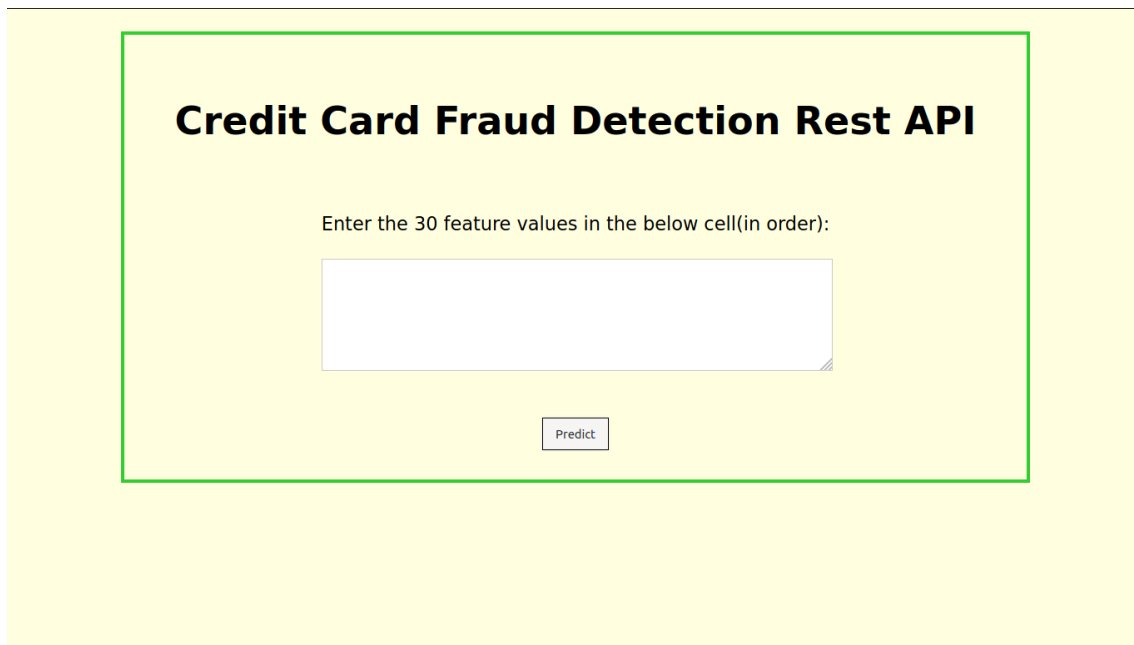


- App `app.py` is the main application file which will run our app, it will be the first file which will be called and executed, after that, this will render the templates, css styles and will also take input given by the user.
- Requirements `requirements.txt` file will contain all the requirements we will be going to use for our work, i.e. model prediction, reading files, reading model files etc.
- Templates `templates` directory contains the html pages we are going to render upon.
- Static `static` contains the css styles which are used for the better formatting of the web app.
- Model `model.pkl` contains the model we got as a result of training which we discussed in the previous section.

- Test data `fraud_values.csv` and `valid_values.csv` are the two files which contains the test data regarding the testing of the web app for our users.
- Procfile `Procfile` is the one file which is most important and mandatory if we want to host our app on Heroku, procfile contains the details about the type of app and which file we want to run while starting the app(in our case, it is `app.py` ).

### How to Predict using the Web App

- Home The home page of the web app will look something like this:



The screenshot shows a web application interface with a light yellow background. A green rectangular border encloses the main content area. At the top of this area, the title "Credit Card Fraud Detection Rest API" is displayed in bold black text. Below the title, a text prompt reads "Enter the 30 feature values in the below cell(in order):". Underneath this prompt is a large, empty white rectangular input box. At the bottom center of the green-bordered area, there is a small button labeled "Predict".

What we need to do is just copy the data entry which we want to predict into the box, this box will take 30 float/integer values as input, with at least one whitespace dividing them.

## Credit Card Fraud Detection Rest API

Enter the 30 feature values in the below cell(in order):

-1.1152878574425795	-19.1397328634111	9.28684735978866
-20.134992104854	7.81867331082574	-15.652207677206302
-1.66834770694329	-21.3404780994803	0.6418997811947
-8.55811832780899	-16.6496281595399	4.81815244787108
-9.44531478308794	1.3170562933234098	-7.24346097400378
0.830910291033798	-9.533257050393189	-18.758641147467398
-8.09264877340557	3.32675827497024	0.42720343146936
-2.1826919456095504	0.5205430723666421	-0.7685564151887328
0.6627666383972359	-0.948454306235033	0.12179592582979301

Predict

Now, we are ready to predict if the transaction entry we provided is fraud or not.

If the transaction will be a fraud transaction, the screen will be something like this:

## Credit Card Fraud Detection Results

Validating provided transaction...

**According to our model, this transaction is a Fraud transaction.**

And if our transaction is not fraud, it will look something like the following:

## Credit Card Fraud Detection Results

Validating provided transaction...

**According to our model, the provided transaction is not a Fraud transaction.**

## Conclusion

Credit Card is a great tool to pay money easily, but as with all the other monetary payment tools, reliability is a issue here too as it is subjected to breach and other frauds. To encounter this problem, a solution is needed to identify the patterns in the transactions and identify the ones which are fraud, so that finding such transactions beforehand in future will be very easy.

Machine Learning is a great tool to do this work since Machine Learning helps us in finding patterns in the data. Machine Learning can help producing great results if provided enough amount of data. Also, with further advances in the technology, Machine Learning too will advance with time, it will be easy for a person to predict if a transaction is fraud or not much more accurately with the advances.

## References

[Web App](#)

[Credit Card Fraud Detection Repository -- Github](#)

[Model training Repository](#)