Credit Card Fraud Detection

Made by:
Aida Himmiche
Michael Asante

Introduction

The purpose of this project is to build an application which operates in the domain of banking or financial organizations, in order to detect fraud in a list of credit card transactions made by a customer.

This application was implemented using:

- **Supervised Machine Learning:** Classification with Random Forest.
- **Weka:** Cross-Validation and analysis
- **Python:** Preprocessing, implementation of the data mining part, and the application backend.
- **Flask:** The Python webapp framework to integrate the ML code.

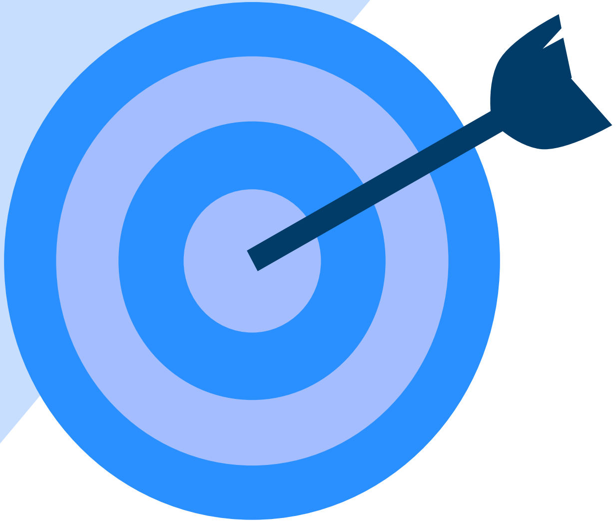
Table of contents

01
KDD Process
Purpose of the application
KDD Process steps

02
Implementation
Python implementation of the
KDD Process

03
Anomaly Detection
Discussion about Anomaly
Detection

04
Application
Web application with
integrated ML classification



01

KDD Process

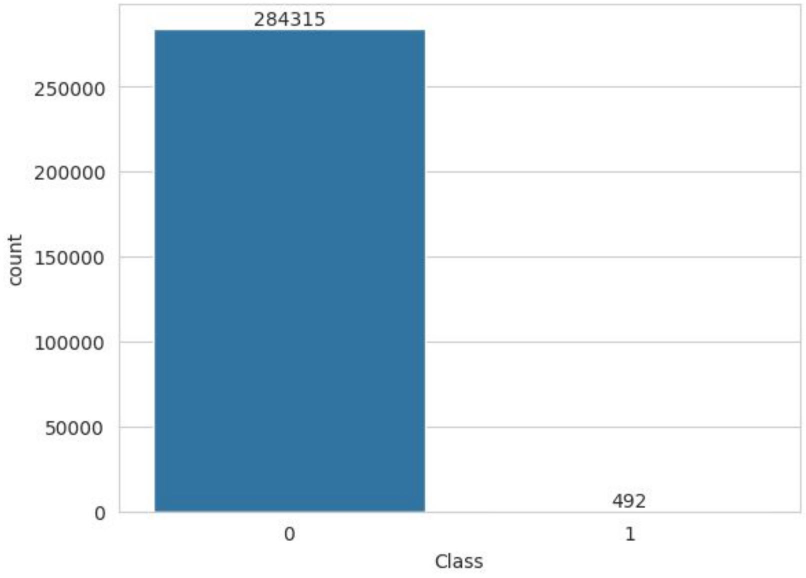
The Dataset

- The dataset used for this project contains 280 000+ transactions from a single credit card.

- It splits the columns into 31 features, 28 of which have been PCA transformed. The rest are Time, Amount, and Class.

- All features contain numerical values .

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V28	Amount	Class
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.021053	149.62	0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	0.014724	2.69	0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	-0.059752	378.66	0
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	0.061458	123.50	0
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	0.215153	69.99	0



The Class Distribution

The dataset is extremely imbalanced:

284 315: normal class

491: fraud class

Preprocessing

The preprocessing was done in three steps:

- **Duplicate Removal:** The number of duplicates found was 1081. Dropping the examples to a count of 283 726.

- **Missing values:** There were no null/missing values in this dataset.

- **Normalization:** The values of the attributes “Time” and “Amount” were transformed using a “RobustScaler” into values in the range (-1, 1) to follow the rest of the features V1 through V28

- **Data Split:** The dataset was split into training and testing data using a stratified holdout method, the percentages used were 66% and 34% respectively

Cross-Validation

This step was carried out to compare different model performances on this data.

To make sure the models have learned patterns and not the data itself, it is necessary to test them against unseen data.

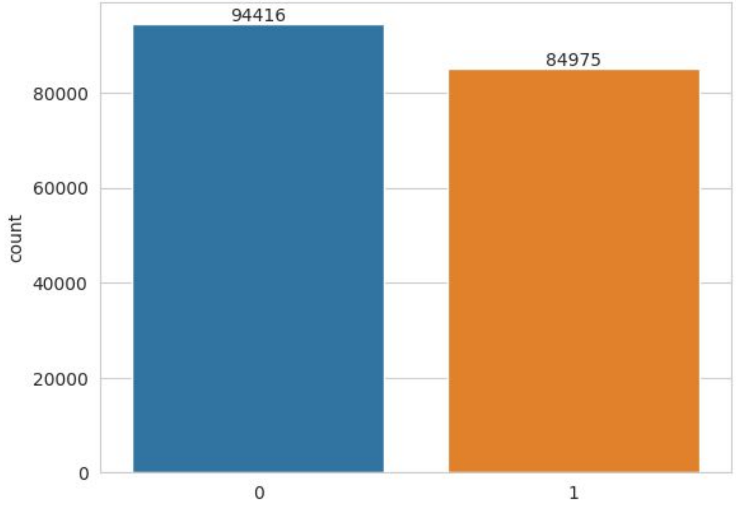
For this we chose the 10-fold cross-validation method with 9 folds for the training set and 1 fold for the validation set.

Data Mining: Rebalancing

To rebalance the training dataset we used a combination of Undersampling and Oversampling that we execute in a Filtered Classifier on WEKA:

- **Undersampling:** used for the majority class “Normal” which has 99.8% support, by applying the “RandomUnderSampler” class from the imblearn package with a sampling strategy of 0.9

- **Oversampling:** used for the minority class “Fraud” which has a bit over 1% support in the entire dataset. We used SMOTE not to train our model on duplicate samples. The sampling strategy was of 0.4

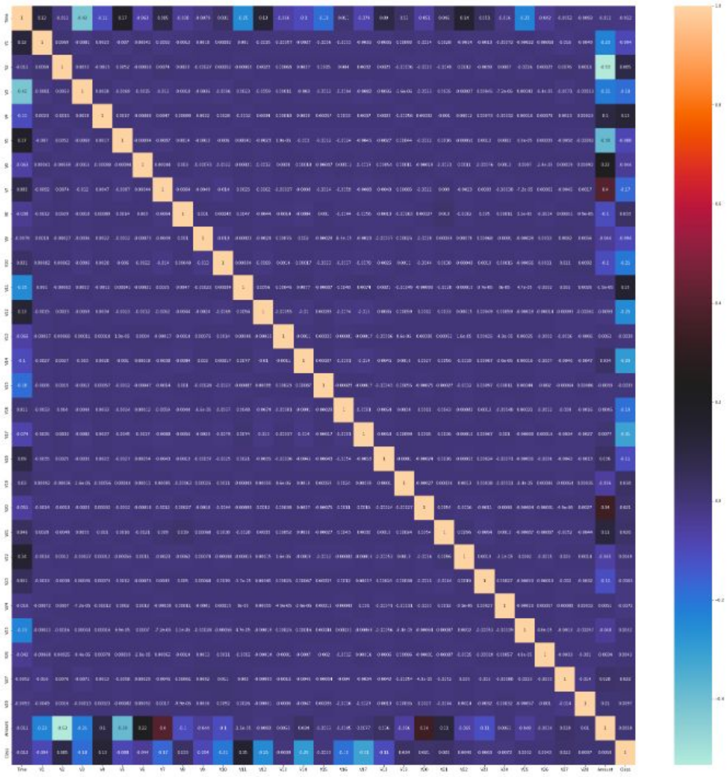


Data Mining: Attribute Selection

Looking at the correlation matrix, we can see that only about half the attribute have a significant correlation with the class.

Using the AttributeSelectedClassifier + FilteredClassfier nested filters on Weka, we were able to try different algorithms for attribute selection namely:

- **CfsSubsetEval + BestFirst:** 8 features: [V3, V4, V10, V11, V12, V14, V16, V17]
- **CorrelationAttributeEval + Ranker:** Similar to the first; it placed the same selected attributes in the top 8 except V2, instead replacing it with V9 as it calculated the Pearson correlation to be higher.
- **InfoGaintAttributeEval + Ranker:** Evaluates the worth of an attribute by measuring the information gain with respect to the class.



Classification: Model Evaluation

The following evaluation measures were considered for this step

Accuracy: Correctly classified per total. TP+TN/Total

Precision: How many predictions of a class actually belong in the class. TP/TP+FP

Recall: How many instances were correctly classified per total instances of that class. TP/TP+FN

F-measure: Arithmetic Mean of Precision and Recall, the higher the better. F = 2 x PR/P+R

Classification: Model Evaluation

Classifier	Method	Selection	%	Precision %	Recall %	F-Measure %	Precision %	Recall %	F-Measure %	Precision %	Recall %	F-Measure %
Random Forest	10-fold CV	-	99.92	100	99.9	100	75.7	85.3	80.1	99.9	99.9	99.9
C45 Pruned	10-fold CV	-	99.23	100	99.3	99.6	16.2	83.8	27.1	99.8	99.2	99.5
C45 Unpruned	10-fold CV	-	99.22	100	99.3	99.6	16.1	83.8	27.0	99.8	99.2	99.5
logistic Function	10-fold CV	-	98.05	100	98.1	99.0	7.4	90.0	13.6	99.8	98.1	98.9
AdaBoost	10-fold CV	-	97.63	100	97.7	98.8	6.0	87.8	11.3	99.8	97.6	98.7
Naive Bayes	10-fold CV	-	97.57	100	97.6	98.8	5.08	86.6	10.9	99.8	97.6	98.6
RandomTree	10-fold CV	-	99.05	100	99.1	99.5	13.5	83.8	23.2	99.8	99.1	99.4
Random Forest	10-fold CV	CfSubsetEval + BestFirst	98.04	100	99.6	99.8	27.6	85.6	99.8	99.9	99.6	99.7
Random Forest	10-fold CV	CfSubsetEval + GreedyStepWise	99.25	100	99.3	99.6	17.0	85.7	28.5	99.8	99.3	99.5
Random Forest	10-fold CV	CorrelationAttributeEval + Ranker	99.9	100	99.9	100	75.9	85.1	80.1	99.9	99.9	99.9
Random Forest	10-fold CV	InfogainAttributeEval+Ranker	98.089	100	99.9	100	75.9	85.4	80.2	99.9	99.9	99.9

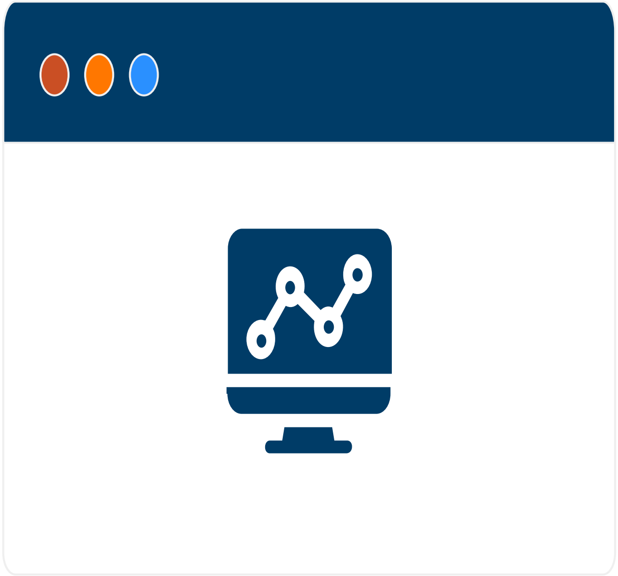
Model Selection

Random Forest

This model performed the best in terms of precision, recall, and F-measure.

White other models only performed well on the “Normal” majority class, the random forest classifier also recognized above 70% of the “Fraud” minority class samples.

No further tests were carried out as the difference in performance was rather clear.



02

Implementation

Random Forest Model

Building the RF Model

```
# Train the model on training data
%time
model = RandomForestClassifier(n_estimators=80,
                               verbose=2,
                               n_jobs=2,
                               oob_score= True,
                               class_weight = "balanced").fit(rebalanced_features, rebalanced_labels)
```

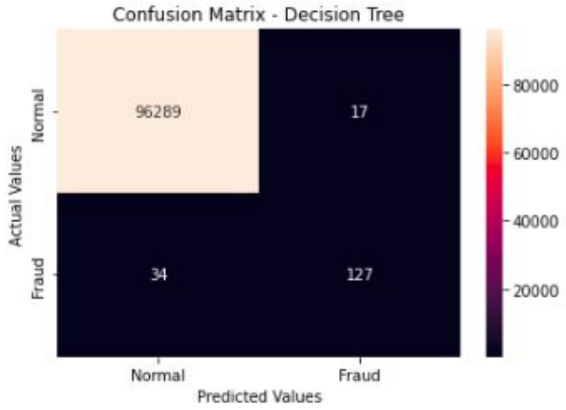
Testing with unseen data

Random Forest achieved 99.9% accuracy due to the high number of examples from the "Normal" class.

Only 17 out of 96289 samples from Normal have been misclassified.

34 out of 127 samples from the "Fraud" class have been misclassified, which means 26%

	Metrics	Results
0	Accuracy	0.999471
1	Precision	0.881944
2	Recall	0.788820
3	F1_score	0.832787



03

Anomaly Detection

Unsupervised Learning

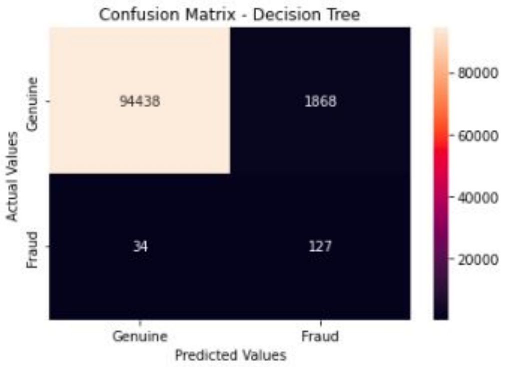
Isolation Forest

Orthogonal space splits + High anomaly score to fewest required “isolation” splits.

```
Prediction Distribution: Counter({0: 94472, 1: 1995})
Errors:  1902
Accuracy Score:
0.9802834129806047
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     96306
           1       0.06      0.79      0.12       161

    accuracy                           0.98     96467
   macro avg       0.53      0.88      0.55     96467
weighted avg       1.00      0.98      0.99     96467
```



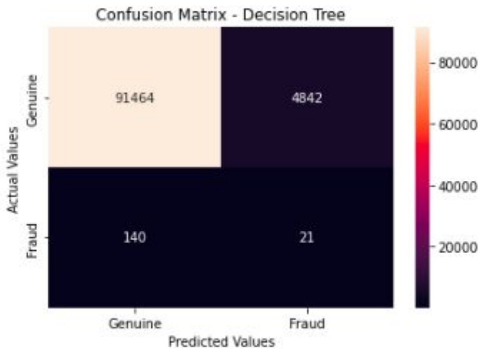
Local Outlier Factor (LOF)

Computes the local density deviation + Outliers are the points that have a substantially lower density than their neighbors.

```
Errors:  4982
Accuracy Score:
0.9483553961458323
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.95      0.97     96306
           1       0.00      0.13      0.01       161

    accuracy                           0.95     96467
   macro avg       0.50      0.54      0.49     96467
weighted avg       1.00      0.95      0.97     96467
```



Unsupervised Learning

One-Class Support Vector Machine (SVM)

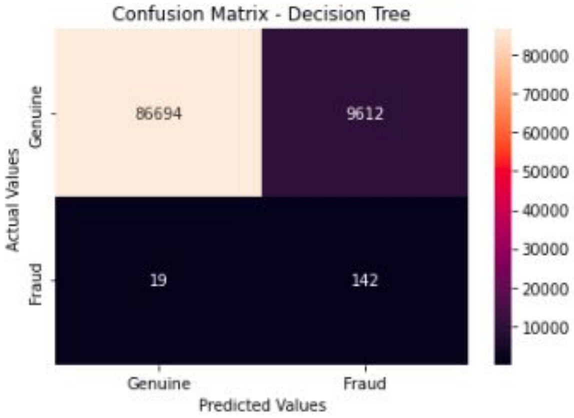
The support vector machine algorithm finds a hyperplane in an N-dimensional space that distinctly classifies the data points using the largest possible margin.

The one class SVM uses a (smallest possible) hypersphere

```
Errors:  9631
Accuracy Score:
0.9001627499559435
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.90      0.95     96306
           1       0.01      0.88      0.03       161

    accuracy                           0.90     96467
   macro avg       0.51      0.89      0.49     96467
weighted avg       1.00      0.90      0.95     96467
```



Reflections...

Even if Anomaly Detection may sound more appropriate for this kind of problem, the performance compared to supervised learning was not impressive.

This method could be useful in the case of non-availability of labeled data, notably the Isolation Forest model.

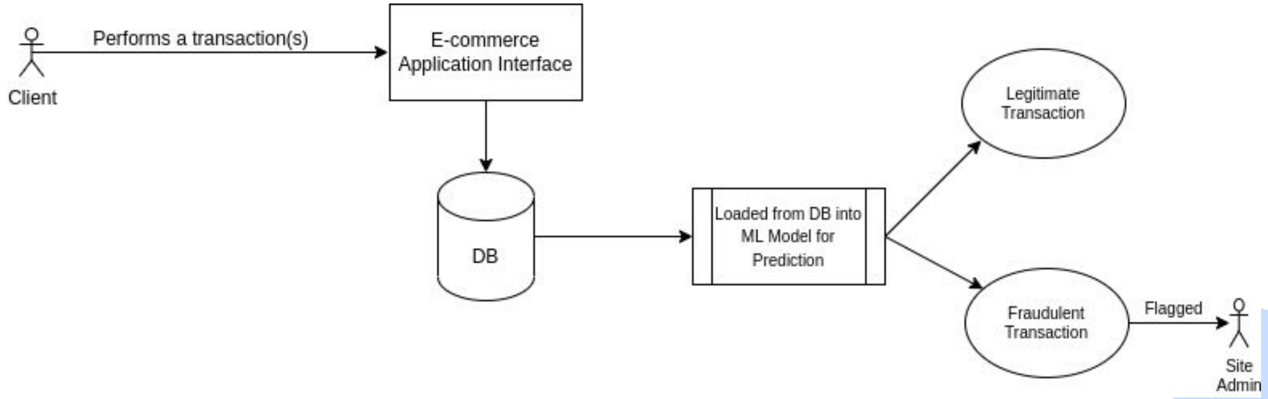
03

Application

Purpose of the Application

This application was designed to serve as a service used by financial organizations in order to determine whether the transactions of a customer are fraudulent or not.

We can illustrate a version of this goal in the following diagram:



Design of the Application

To simplify the scope of the web application, we described the functional and non-functional requirements as follows:

Functional:

- The user may input a list of transactions through the platform in a ".csv" format
- They may see their inputted transactions on the web page.
- They may use the model to predict the class of each transaction.

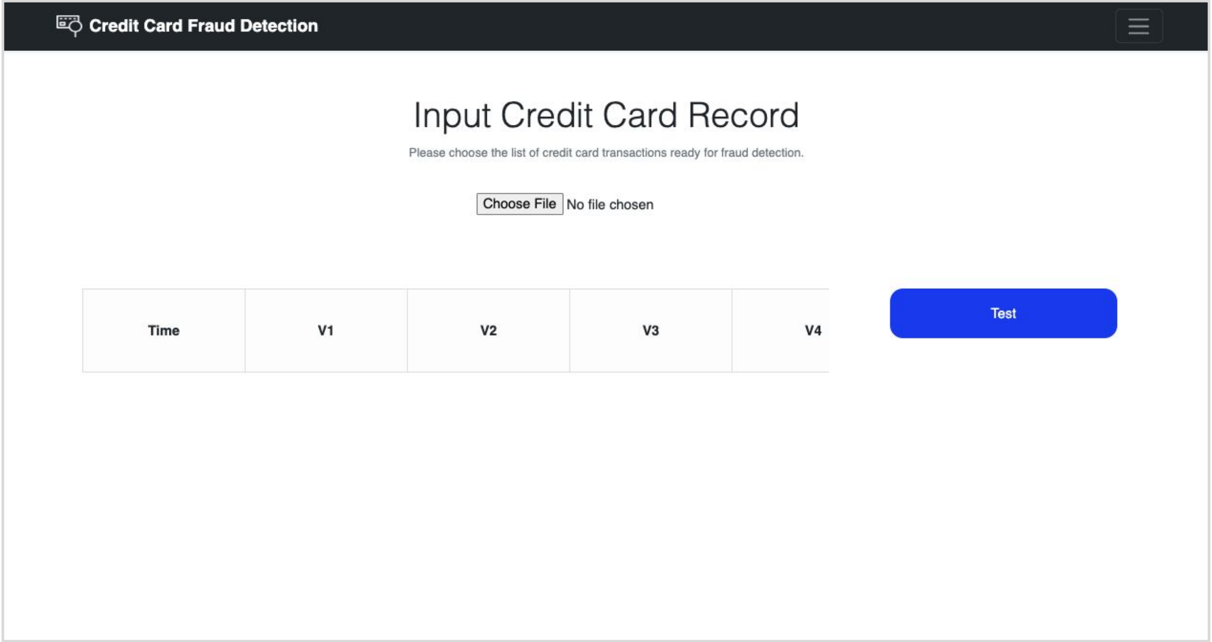
Non-functional:

- The app shall be easy to use
- The user shall not wait long for any of the functionalities above.
- The model must predict at best accuracy and reduce false positives/false negatives.

Testing the Application

The front page:

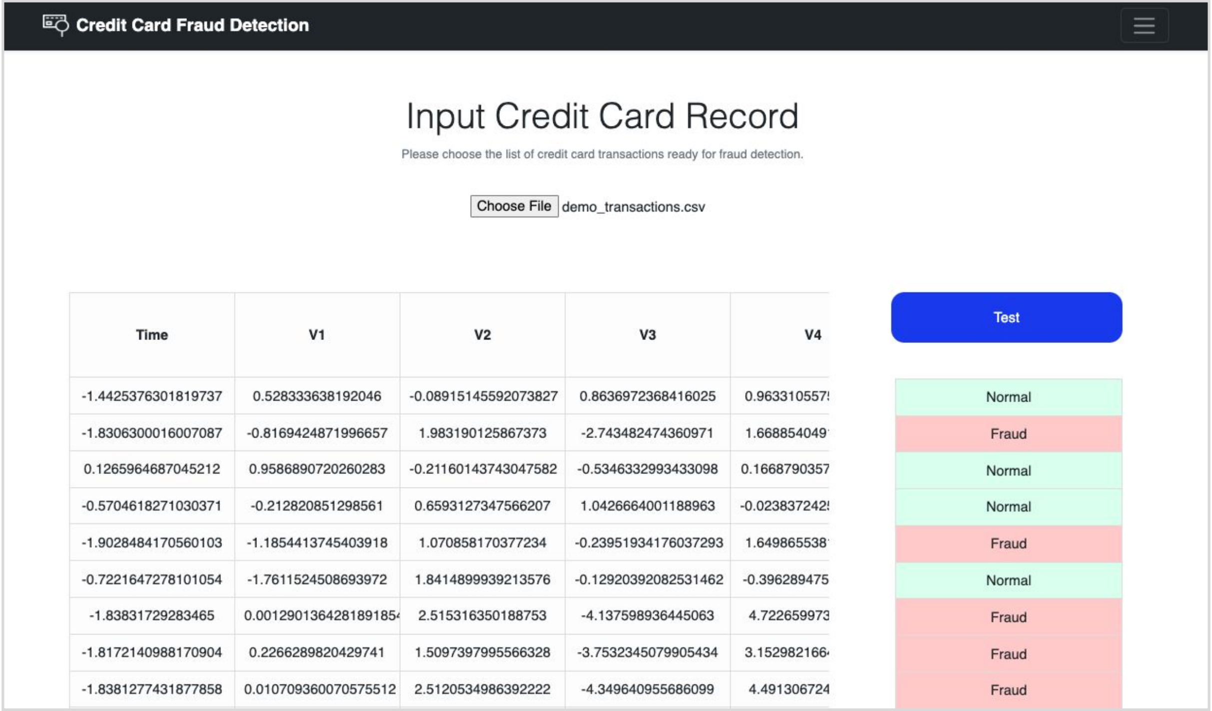
Allows a user to choose their preferred file. (List of transactions)



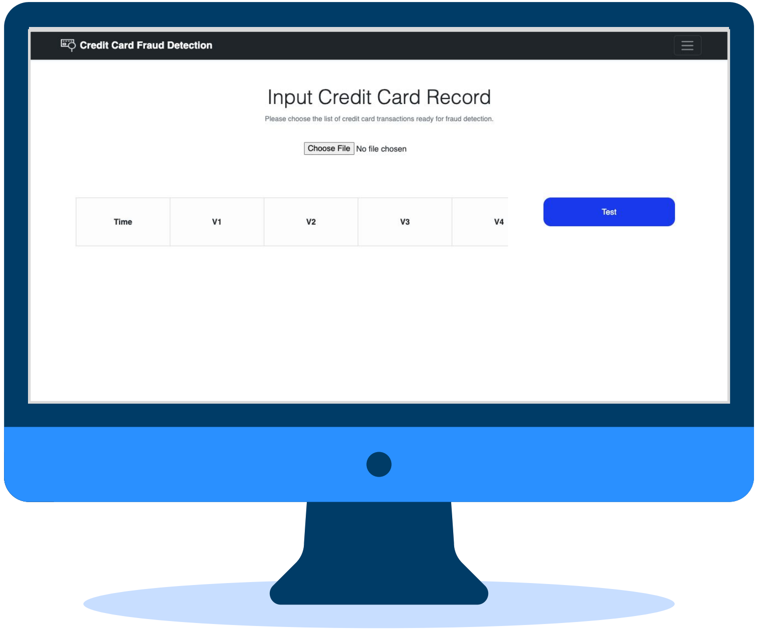
Testing the Application

The prediction results:

What the user sees after clicking the “Test” button.



Time	V1	V2	V3	V4	
-1.4425376301819737	0.528333638192046	-0.08915145592073827	0.8636972368416025	0.963310557	Normal
-1.8306300016007087	-0.8169424871996657	1.983190125867373	-2.743482474360971	1.668854049	Fraud
0.1265964687045212	0.9586890720260283	-0.21160143743047582	-0.5346332993433098	0.1668790357	Normal
-0.5704618271030371	-0.212820851298561	0.6593127347566207	1.0426664001188963	-0.023837242	Normal
-1.9028484170560103	-1.1854413745403918	1.070858170377234	-0.23951934176037293	1.649865538	Fraud
-0.7221647278101054	-1.7611524508693972	1.8414899939213576	-0.12920392082531462	-0.396289475	Normal
-1.83831729283465	0.001290136428189185	2.515316350188753	-4.137598936445063	4.722659973	Fraud
-1.8172140988170904	0.2266289820429741	1.5097397995566328	-3.7532345079905434	3.152982166	Fraud
-1.8381277431877858	0.010709360070575512	2.5120534986392222	-4.349640955686099	4.491306724	Fraud


Credit Card Fraud Detection
Input Credit Card Record
Please choose the list of credit card transactions ready for fraud detection.
Choose File No file chosen
Time
V1
V2
V3
V4
Test

Live Demo!

Thanks!

Do you have any questions?