U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Supervised Learning

Carlos Eduardo Coelho Veríssimo - 201907716

Frederico Manuel Lopes - 201904580

Nuno Miguel Carvalho de Jesus -201905477

Group 05_1A

Work specification

The selected theme was Credit Card Fraud.

We were given a Dataset containing various details such as “distance from home” and “online order”, regarding credit card transactions. It contains 1 million entries, of which 87403 are labeled as fraudulent.

The dataset also has a boolean column called ‘Fraud’ that labels whether or not a transaction was fraudulent and other attributes such as “ratio to median purchase price”, “online order”, “distance from last transaction” and others.

The goal of this project is to train several models to be able to recognize if a given transaction is “normal” or “fraud” based on its attributes.

Related work

- Machine Learning with Neural Networks Using scikit-learn | Pluralsight
- Visualization with Seaborn | Python Data Science Handbook
- Sklearn
- Credit Card Fraud Detection Github

Tools and Algorithms

The selected language for development was **Python**.

The development environment used was **Jupyter Notebook.**

Used Tools:

- **matplotlib -** to create the charts
- **pandas** - to analyse and prepare the data
- **sklearn** - to create, train and the test the performance of the models
- **imblearn** - to balance the dataset

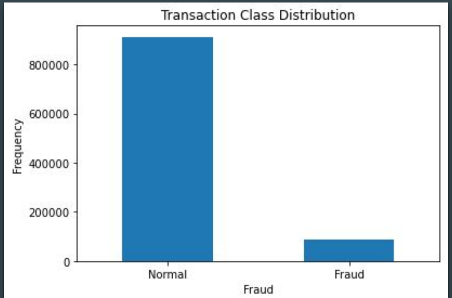
Implemented Models:

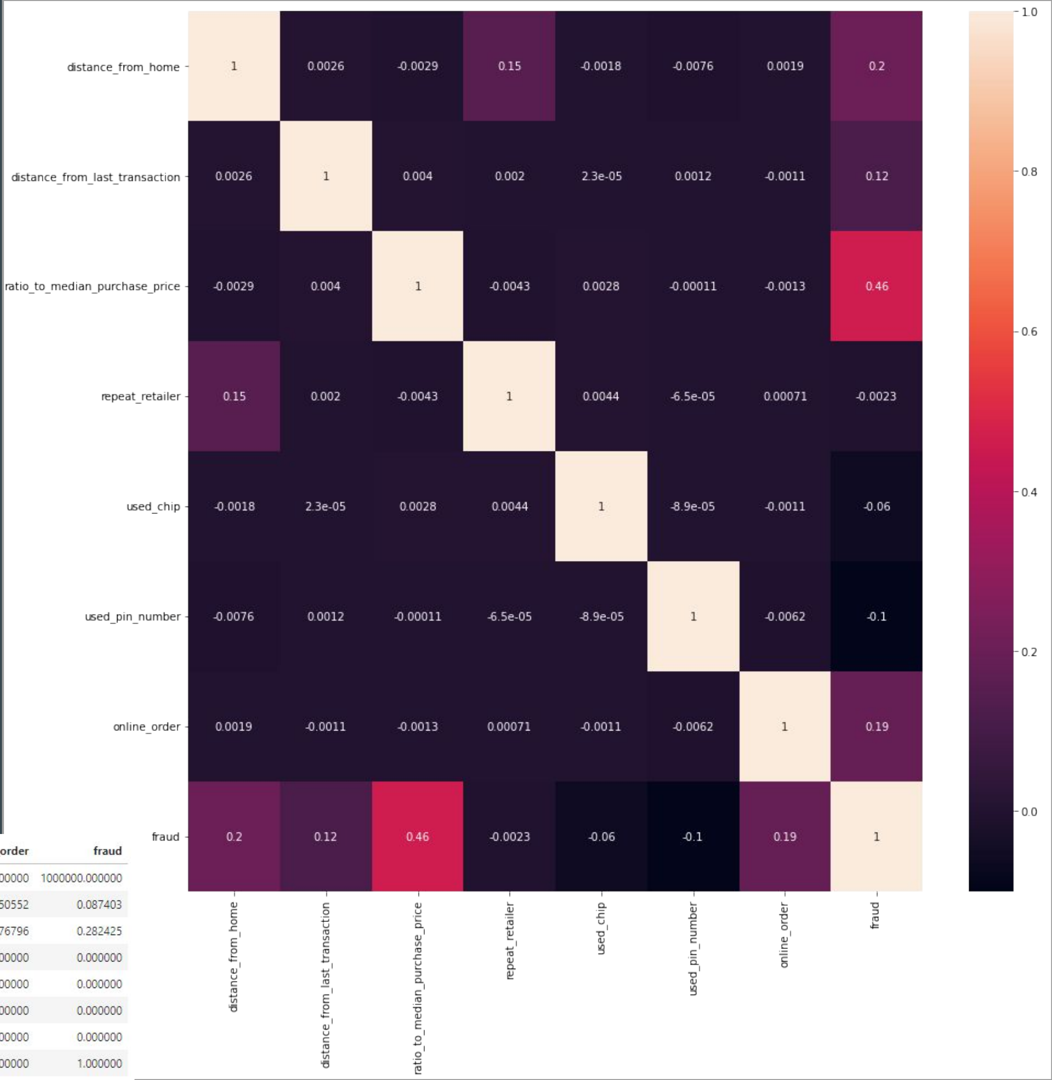
- **Decision Tree,**
- **Neural Networks,**
- **K-NN**
- **SVM**

Data pre-processing(1/2)

Data analysis:

- We began by checking the data columns and by making sure there were no missing values.
- After making sure that the data was ok, we now needed to have an idea of the number of normal and fraudulent transactions that existed in the data
- Finally, we created a correlation matrix to check the most influential feature on the “fraud” value





	distance_from_home	distance_from_last_transaction	ratio_to_median_purchase_price	repeat_retailer	used_chip	used_pin_number	online_order	fraud
count	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000	1000000.000000
mean	26.628792	5.036519	1.824182	0.881536	0.350399	0.100608	0.650552	0.087403
std	65.390784	25.843093	2.799589	0.323157	0.477095	0.300809	0.476796	0.282425
min	0.004874	0.000118	0.004399	0.000000	0.000000	0.000000	0.000000	0.000000
25%	3.878008	0.296671	0.475673	1.000000	0.000000	0.000000	0.000000	0.000000
50%	9.967760	0.998650	0.997717	1.000000	0.000000	0.000000	1.000000	0.000000
75%	25.743985	3.355748	2.096370	1.000000	1.000000	0.000000	1.000000	0.000000
max	10632.723672	11851.104565	267.802942	1.000000	1.000000	1.000000	1.000000	1.000000

Data pre-processing (2/2)

Data preparation:

- Separation of labels and features
- Division of Dataset into training and test data (25% / 75%)
- Creation of balanced dataset by undersampling the “normal” transactions
- Creation of data only containing the positive correlation attributes
- Creation of a smaller dataset for the training of SVM (1% / 99%)
- Creation of a several datasets with different percentages of training and test data to see the evolution of the accuracy and F1
- Cross validation (k=10)

Developed models (1/4)

Decision Tree:

Balanced Dataset:

Confusion matrix:

[[684264   186]

[7 65543]]

F1: 0.9985298486429665

Accuracy: 0.9997426666666667

Imbalanced Dataset (with all features):

Confusion matrix:

[[684436   14]

[28 65522]]

F1: 0.9996795996521368

Accuracy: 0.999944

Imbalanced Dataset (selected features):

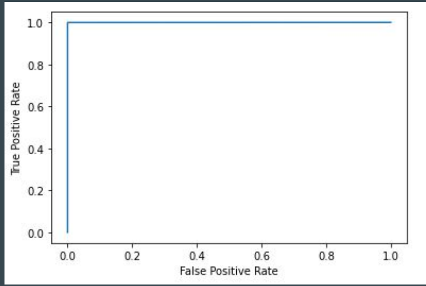
Confusion matrix:

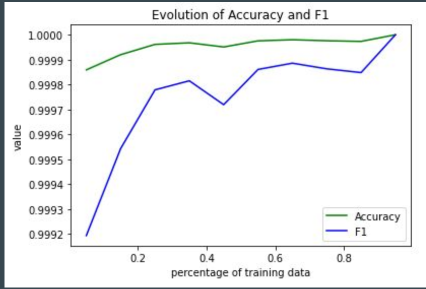
[[672484 11966]

[12418 53132]]

F1: 0.8133610924009552

Accuracy: 0.96748





Developed models (2/4)

We chose an architecture of (20,20) for the hidden layers since it was what gave us the best results, more layers didn't work better. The biggest difference in accuracies was due to the number of training data.

Neural Networks:

Balanced Dataset:

Confusion matrix:

[[674372 10078]

[110 65440]]

F1: 0.927779510590637

Accuracy: 0.986416

Imbalanced Dataset (with all features):

Confusion matrix:

[[683795   655]

[977 64573]]

F1: 0.9875208368380003

Accuracy: 0.997824

Imbalanced Dataset (selected features):

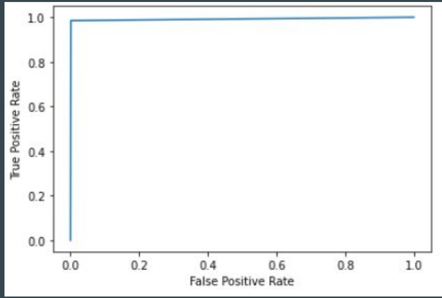
Confusion matrix:

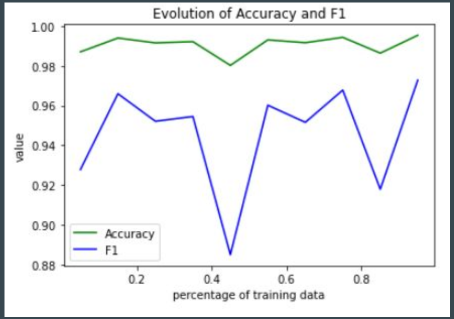
[[669146 15304]

[4161 61389]]

F1: 0.8631567106992962

Accuracy: 0.9740466





Cross-validation(k=10):

Accuracy: 0.998247

Developed models (3/4)

We chose 3 neighbours since it was what gave us the best results. 2, 5 and 7 was similar but worst.

K-NN:

Balanced Dataset:

Confusion matrix:

[[633849 50601]

[516 65034]]

F1: 0.7178739961917378

Accuracy: 0.931844

Imbalanced Dataset (with all features):

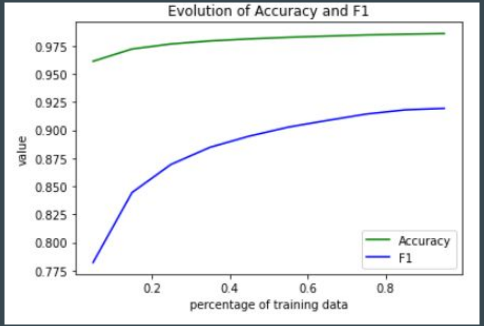
Confusion matrix:

[[674270 10180]

[7298 58252]]

F1: 0.8695496409965517

Accuracy: 0.976696



Imbalanced Dataset (selected features):

Confusion matrix:

[[668637 15813]

[12868 52682]]

F1: 0.7860345406393374

Accuracy: 0.9617586666666667

Developed models (4/4)

We only used 1% of the data (10000 transactions) as training data since the training took too long for larger values but still achieved decent results

SVM:

Balanced Dataset:

Confusion matrix:

[[831992 71476]

[5775 80757]]

F1: 0.6764559294703998

Accuracy: 0.9219686868686868

Imbalanced Dataset (with all features):

Confusion matrix:

[[898060   5408]

[33498 53034]]

F1: 0.731634637935078

Accuracy: 0.9607010101010101

Imbalanced Dataset (selected features):

Confusion matrix:

[[895237   8231]

[46977 39555]]

F1: 0.5889754165487872

Accuracy: 0.9442343434343434

Models Comparison