Fraud Detection

Supervised Machine Learning

Author
KIERAN THAKKAR

Introduction

Online payment systems have resulted in increased payment frauds. Online payment frauds can happen with anyone using any payment system, especially while making payments using a credit card.

Detecting online payment fraud is very important for credit card companies to ensure that the customers are not getting charged for the products and services they never paid.

In this project we will create a model to detect online payment fraud.

Objectives

1. Explore the data – what are the fields, how are they distributed, will they require any transformations?
2. Prepare the dataset
3. Create 3 supervised ML models
4. Measure and compare the accuracies
5. Select best model

Data Exploration

WHAT ARE WE DEALING WITH?

Importing the dataset

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0
5	1	PAYMENT	7817.71	C90045638	53860.0	46042.29	M573487274	0.0	0.0	0

Key data / columns:

step: represents a unit of time where 1 step equals 1 hour

type: type of online transaction

amount: the amount of the transaction

oldbalanceOrg: balance before the transaction

newbalanceOrig: balance after the transaction

isFraud: fraud transaction, (1/0 → yes/no)

Values

- Mix between numerical and string

- There were no null values
- We have a complete dataset

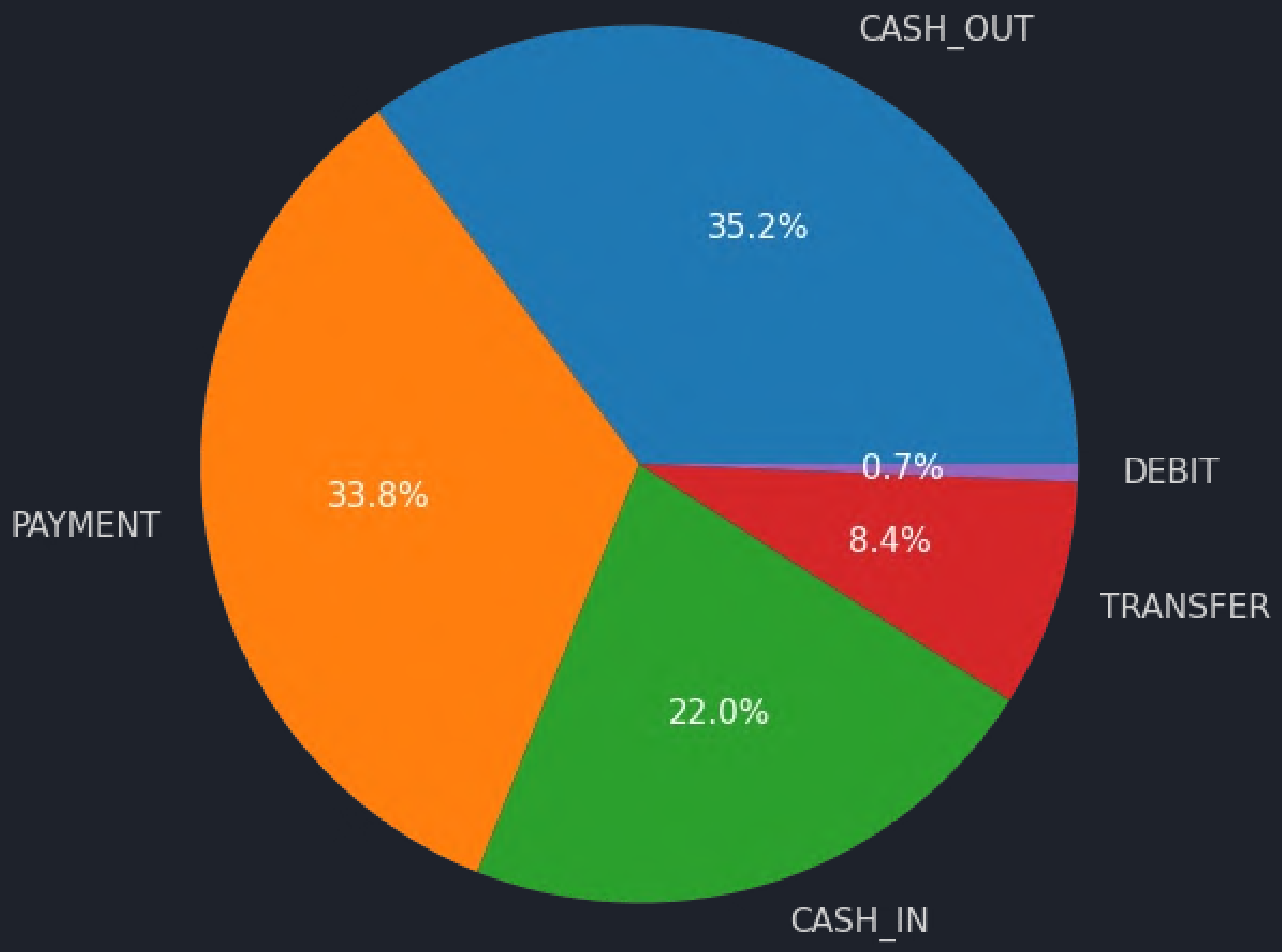
```
df.info()

RangeIndex: 6362620 entries
Data columns (total 11 columns):
 #   Column          Dtype
---  ------          -----
 0   step            int64
 1   type            object
 2   amount          float64
 3   nameOrig        object
 4   oldbalanceOrg   float64
 5   newbalanceOrig  float64
 6   nameDest        object
 7   oldbalanceDest  float64
 8   newbalanceDest  float64
 9   isFraud         int64
 10  isFlaggedFraud  int64
```

```
df.isnull().sum()

step              0
type              0
amount            0
nameOrig          0
oldbalanceOrg     0
newbalanceOrig    0
nameDest          0
oldbalanceDest    0
newbalanceDest    0
isFraud           0
isFlaggedFraud    0
```

TRANSACTION TYPES



Transaction Types

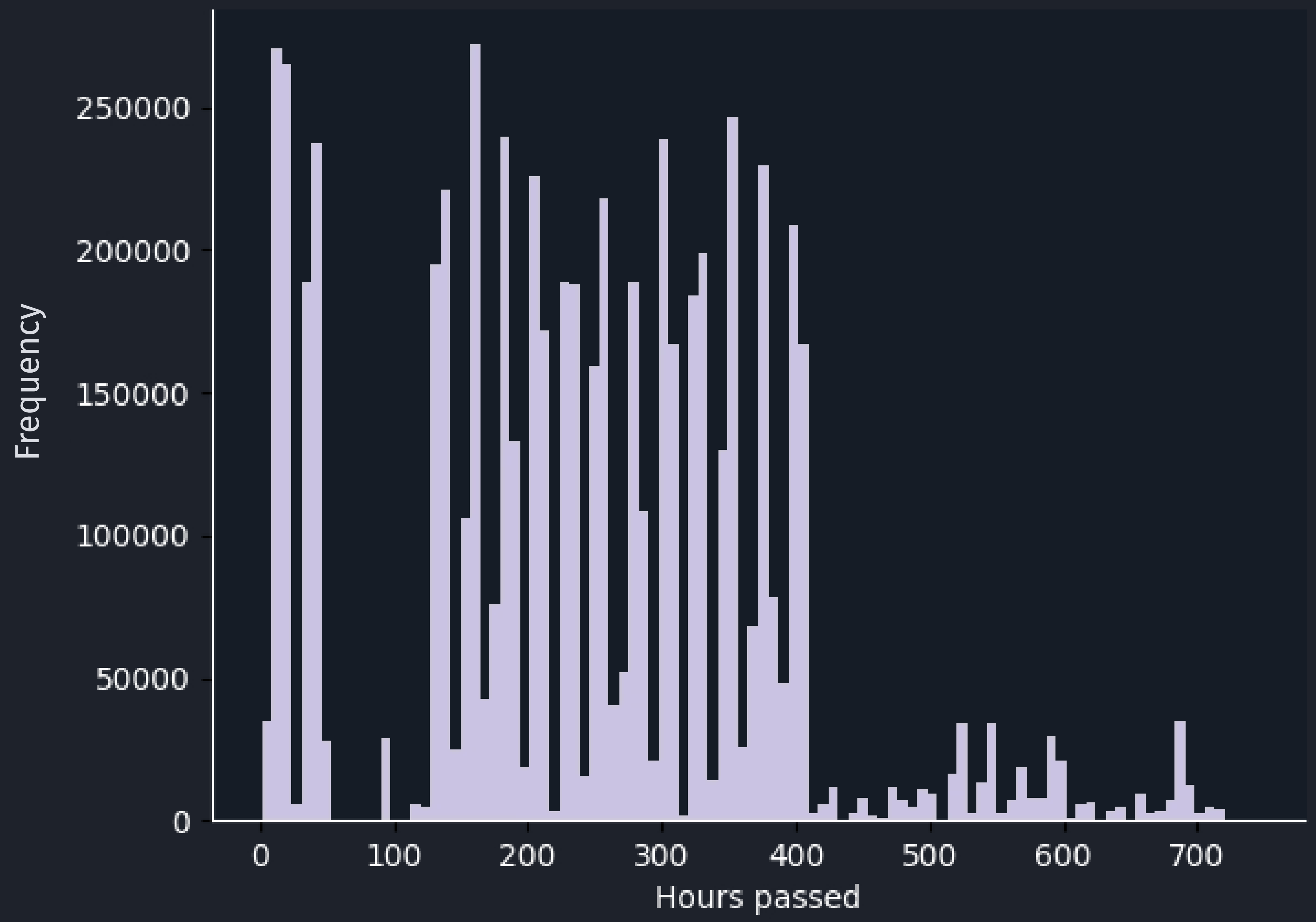
Some background information about the graph: where the numbers come from, what they mean, and why it matters.

TRANSACTION AMOUNT (100 BINS)

Transaction Amounts

Some background information about the graph: where the numbers come from, what they mean, and why it matters.

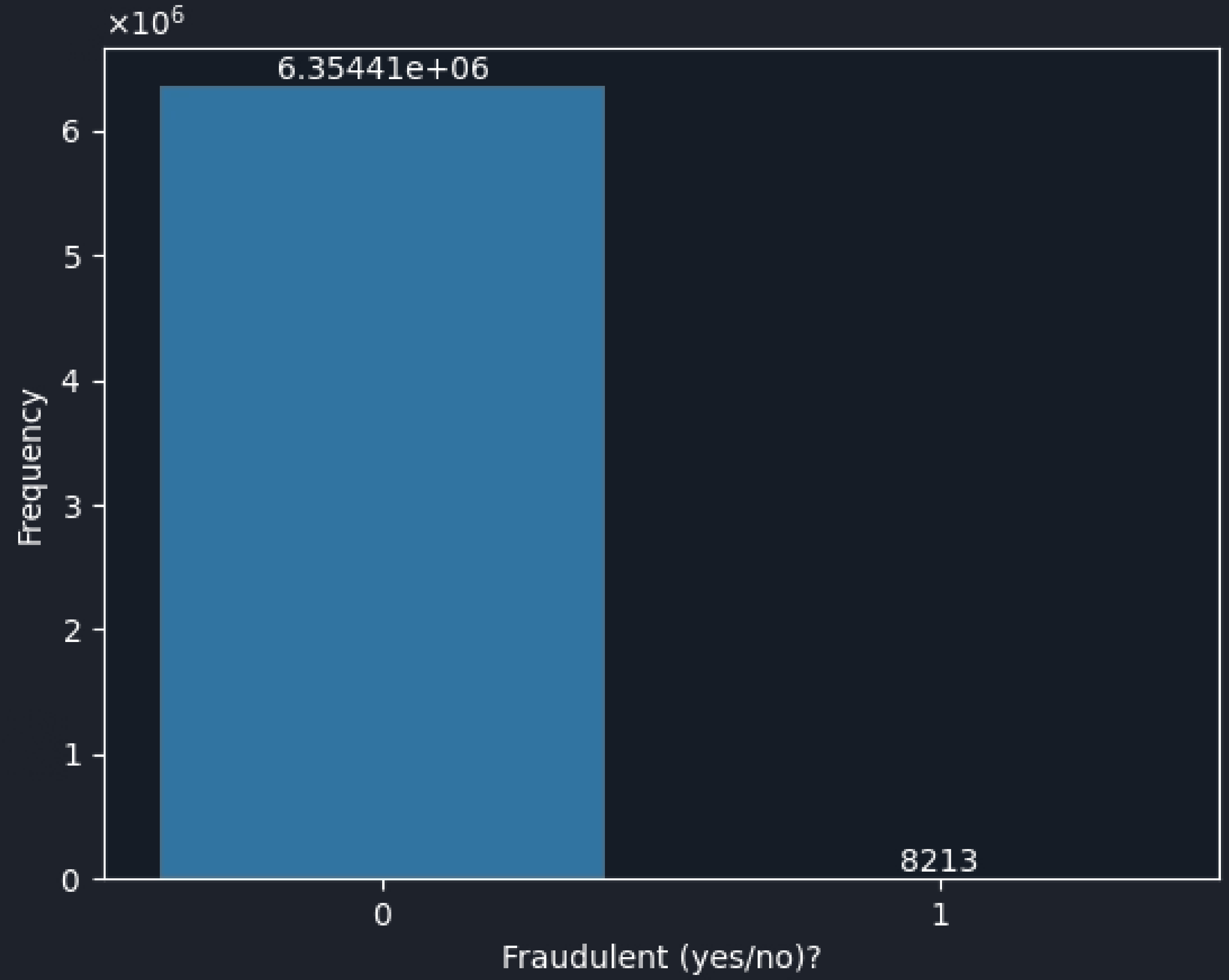
TRANSACTION STEP / TIME (100 BINS)



Transaction Time

Some background information about the graph: where the numbers come from, what they mean, and why it matters.

FRAUDULENT TRANSACTIONS



isFraud

Some background information about the graph: where the numbers come from, what they mean, and why it matters.

Data Preparation

"PREPROCESSING"

Type Mapping

```
type_map = {"PAYMENT": 1, "TRANSFER": 2, "CASH_OUT": 3, "DEBIT": 4, "CASH_IN": 5}
df.type = df.type.map(type_map)
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	1	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	1	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	2	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	3	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	1	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

Drop unwanted columns

```
droppers = ["step", "nameOrig", "nameDest", "isFlaggedFraud"]
df = df.drop(axis=1, droppers)
df.head()
```

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	9839.64	170136.0	160296.36	0.0	0.0	0
1	1	1864.28	21249.0	19384.72	0.0	0.0	0
2	2	181.00	181.0	0.00	0.0	0.0	1
3	3	181.00	181.0	0.00	21182.0	0.0	1
4	1	11668.14	41554.0	29885.86	0.0	0.0	0

- All remaining 'object' columns were dropped.
- These were destination names, millions of values that would not work w/ ML.
- Also dropped step, because fraud can happen at any time.

Drop unwanted columns

```
droppers = ["step", "nameOrig", "nameDest", "isFlaggedFraud"]
df = df.drop(axis=1, droppers)
df.head()
```

	type	amount	oldbalanceOrg	newbalanceOrig	oldbalanceDest	newbalanceDest	isFraud
0	1	9839.64	170136.0	160296.36	0.0	0.0	0
1	1	1864.28	21249.0	19384.72	0.0	0.0	0
2	2	181.00	181.0	0.00	0.0	0.0	1
3	3	181.00	181.0	0.00	21182.0	0.0	1
4	1	11668.14	41554.0	29885.86	0.0	0.0	0

- All remaining 'object' columns were dropped.
- These were destination names, millions of values that would not work w/ ML.
- Also dropped step, because fraud can happen at any time.

Modelling

Results and Comparisons

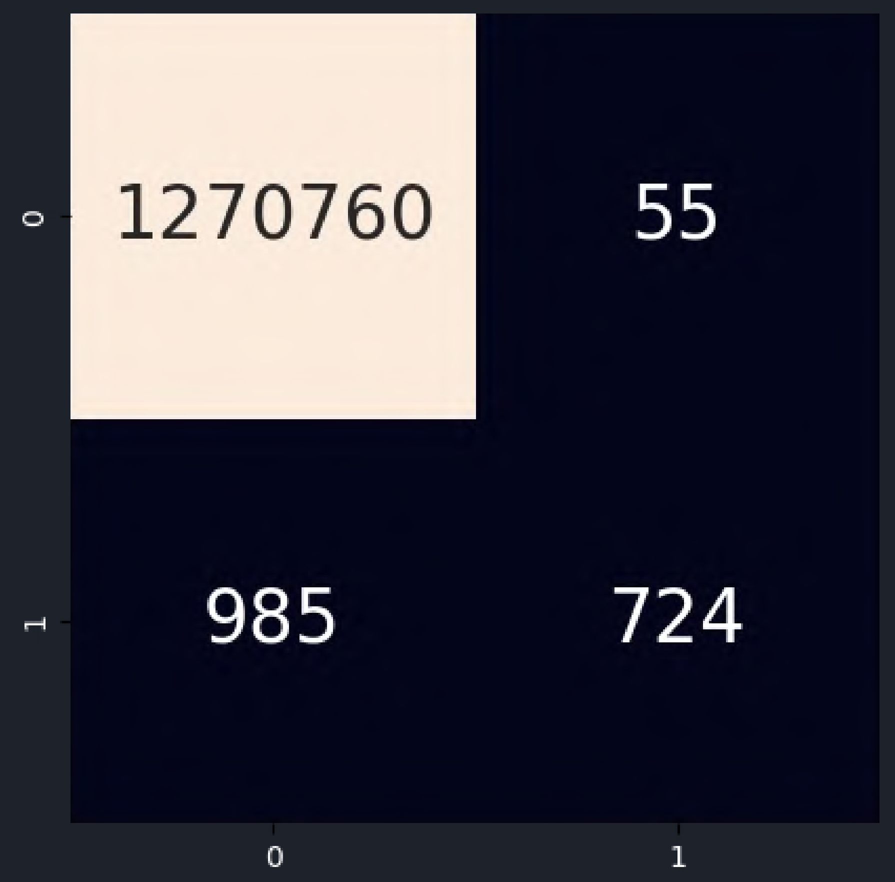
Confusion Matrices

NEURAL NETWORK

Accuracy: 28%

Precision: ~0%

LOGISTIC REGRESSION



Accuracy: 99.92%

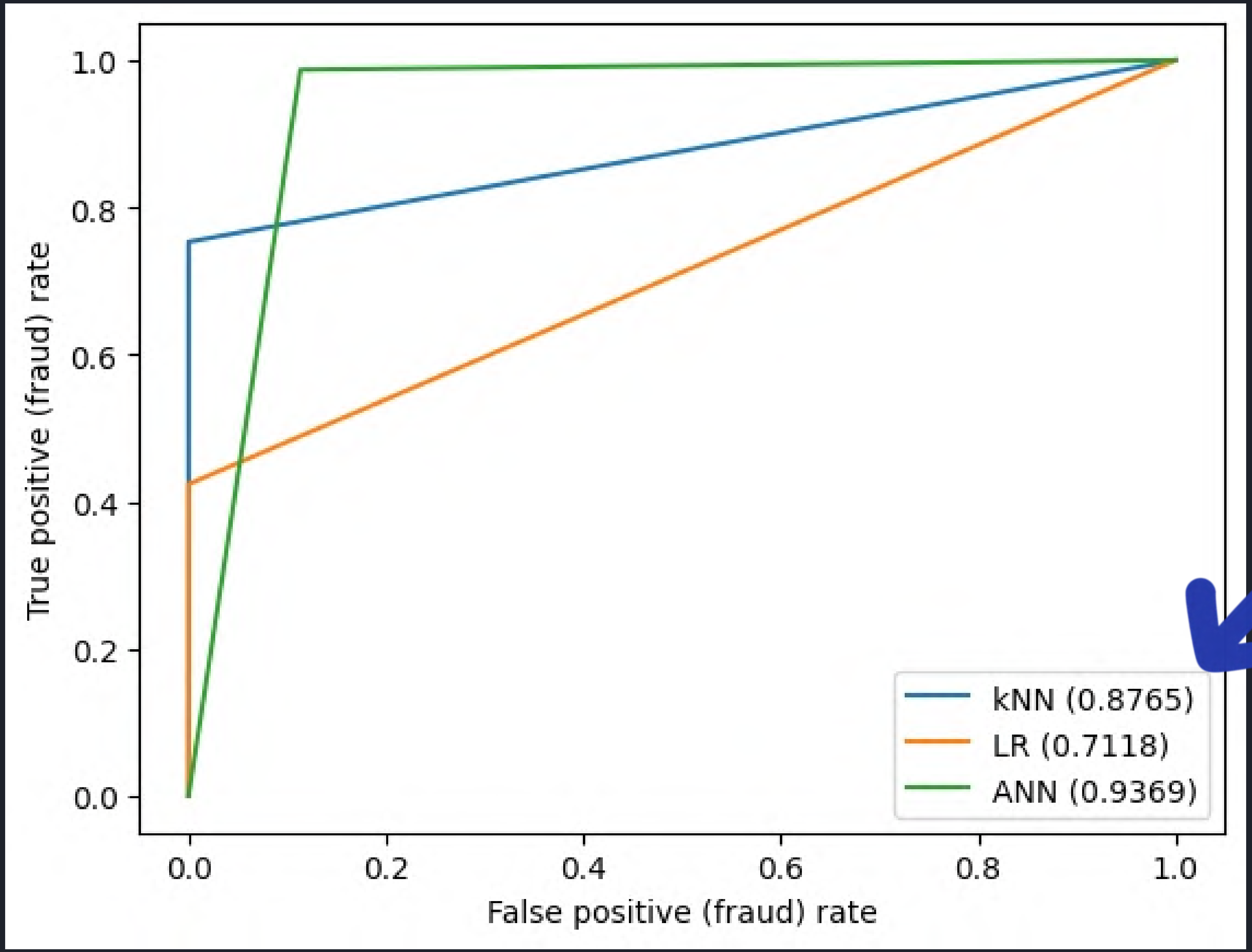
Precision: 93%

K-NEAREST NEIGHBOURS

Accuracy: 99.96%

Precision: 93%

ROC Curve



AUC Scores

Try Pitch

Conclusions

Conclusions

1. kNearestNeighbours was the best overall model
2. Fastest computation time was LogisticRegression()
3. Creating a neural network was not worth it
4. Complexity != Good model
    a. Memory allocation
    b. Computation times
    c. Stress

Thanks for watching!

Pitch

Want to make a presentation like this one?

Start with a fully customizable template, create a beautiful deck in minutes, then easily share it with anyone.

Create a presentation (It’s free)