Credit Card Fraud Prediction

Elaine Li, Terry Mu

Table of Contents

01 Introduction

02 Preprocessing

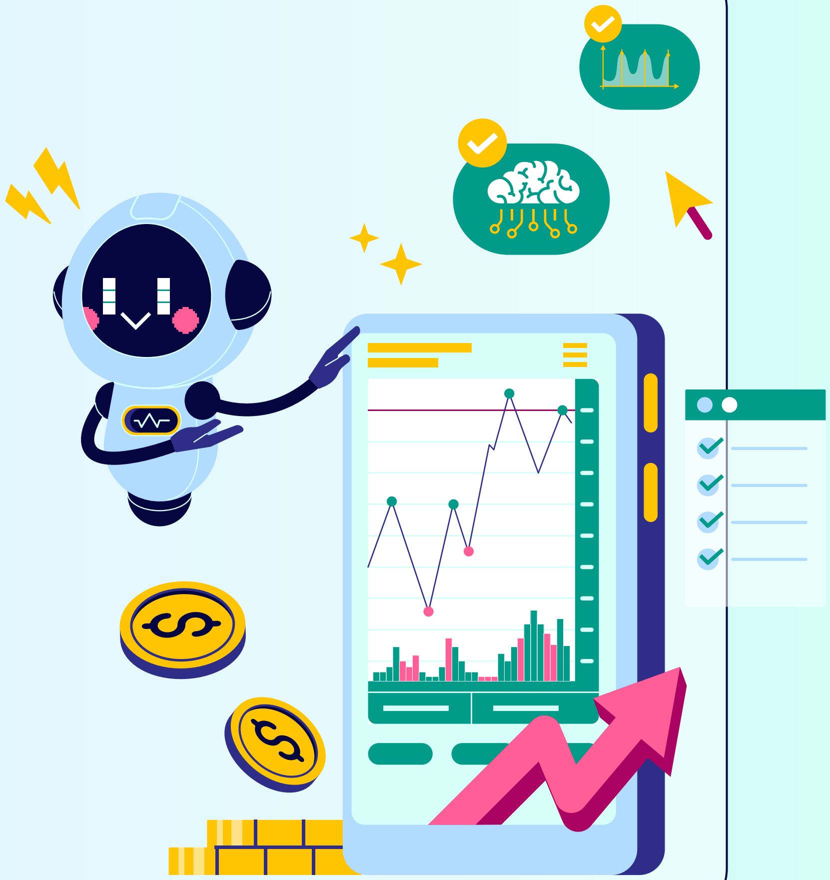
03 Model Training

04 Result

05 Conclusion

01

Introduction



Problem Statement

Classify the fraud (1) and non-fraud (0) credit card

```
Data columns (total 23 columns):
 #   Column                 Non-Null Count   Dtype
---  ------                 --------------   -----
 0   Unnamed: 0             555719 non-null  int64
 1   trans_date_trans_time  555719 non-null  object
 2   cc_num                 555719 non-null  float64
 3   merchant               555719 non-null  object
 4   category               555719 non-null  object
 5   amt                    555719 non-null  float64
 6   first                  555719 non-null  object
 7   last                   555719 non-null  object
 8   gender                 555719 non-null  object
 9   street                 555719 non-null  object
 10  city                   555719 non-null  object
 11  state                  555719 non-null  object
 12  zip                    555719 non-null  int64
 13  lat                    555719 non-null  float64
 14  long                   555719 non-null  float64
 15  city_pop               555719 non-null  int64
 16  job                    555719 non-null  object
 17  dob                    555719 non-null  object
 18  trans_num              555719 non-null  object
 19  unix_time              555719 non-null  int64
 20  merch_lat              555719 non-null  float64
 21  merch_long             555719 non-null  float64
 22  is_fraud               555719 non-null  int64
```

21 Features, 1 target (1=fraud/0=non-fraud)

Dataset Characteristics

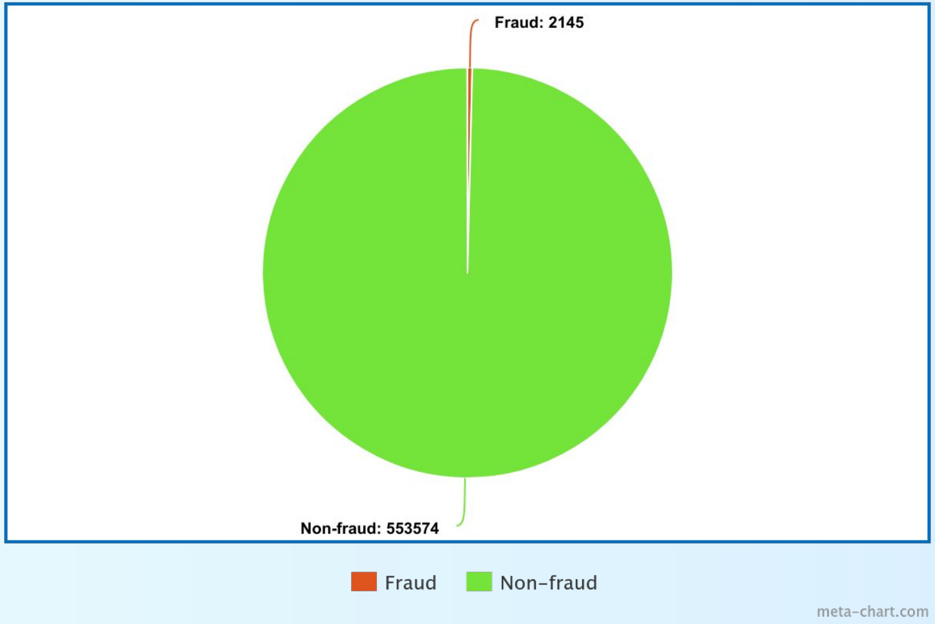
No null values 😋

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

Unnamed: 0               0
trans_date_trans_time    0
cc_num                   0
merchant                 0
category                 0
amt                      0
first                    0
last                     0
gender                   0
street                   0
city                     0
state                    0
zip                      0
lat                      0
long                     0
city_pop                 0
job                      0
dob                      0
trans_num                0
unix_time                0
merch_lat                0
merch_long               0
is_fraud                 0
dtype: int64
```

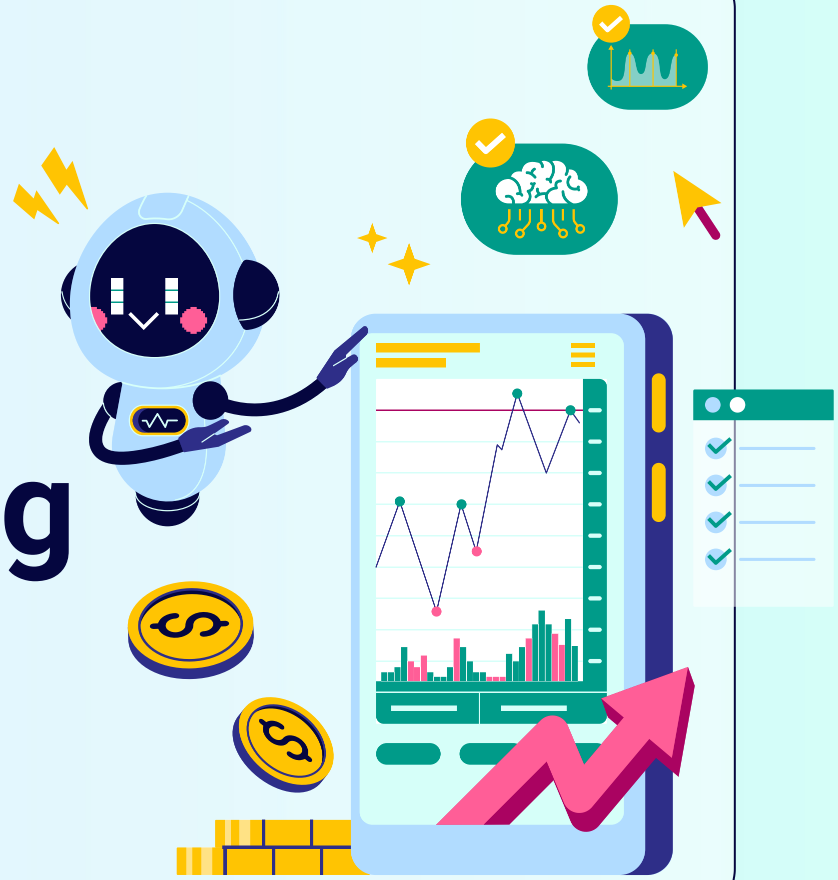
Extremely imbalanced data 😡

Only 0.386% of samples are fraud!



02

Preprocessing



Feature Selection & Encoding

```
h_cardinality = [var for var in obj_features if df[var].nunique() > 100 and var != 'trans_date_trans_time']
h_cardinality
```

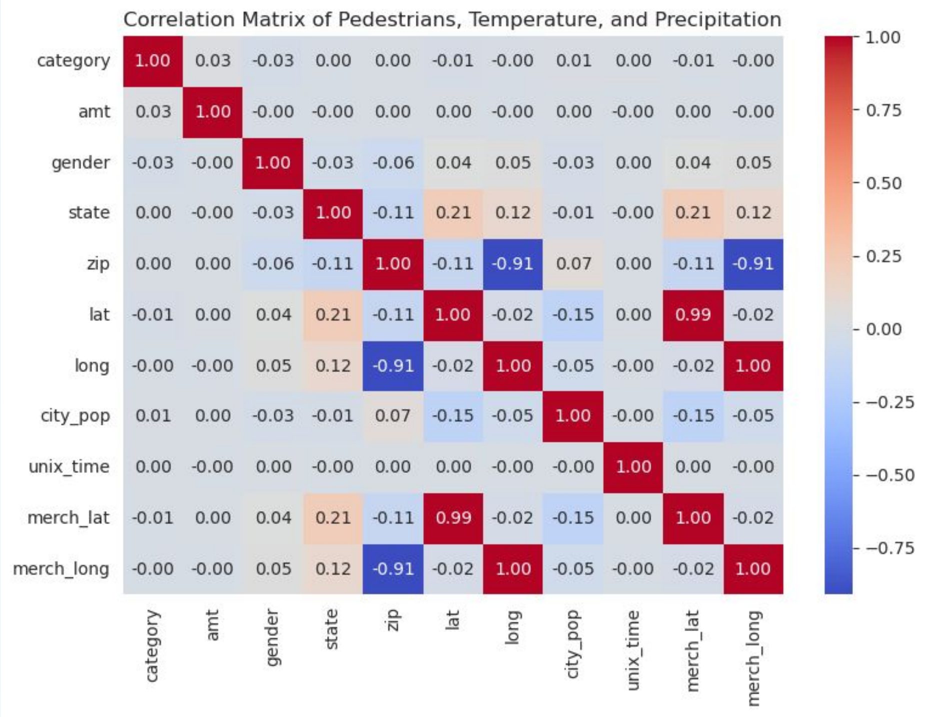
```
['merchant', 'first', 'last', 'street', 'city', 'job', 'dob', 'trans_num']
```

- Check and remove object features with more than 100 unique values

```
X.category = X.category.astype('category').cat.codes
X.gender = X.gender.astype('category').cat.codes
X.state= X.state.astype('category').cat.codes
```

- Encoding the categorical data

Examine duplicate features



Extremely high correlation

- Zip & (merchant) longitude
- Longitude & merchant_longitude
- Latitude & merchant_latitude

Data Scaling

```
scaler=StandardScaler()
scaler.fit(X)
X_train= scaler.transform(X_train)
X_test= scaler.transform(X_test)
X_val = scaler.transform(X_val)
```

Under/Oversampling

```
sampler = RandomUnderSampler(random_state=42)
X_train, y_train = sampler.fit_resample(X_train, y_train)
X_train.shape
```

```
(3090, 8)
```

```
from imblearn.over_sampling import SMOTE

sampler = SMOTE(random_state=42)
X_train, y_train = sampler.fit_resample(X_train, y_train)
```

```
(542558, 45)
```

Undersampling

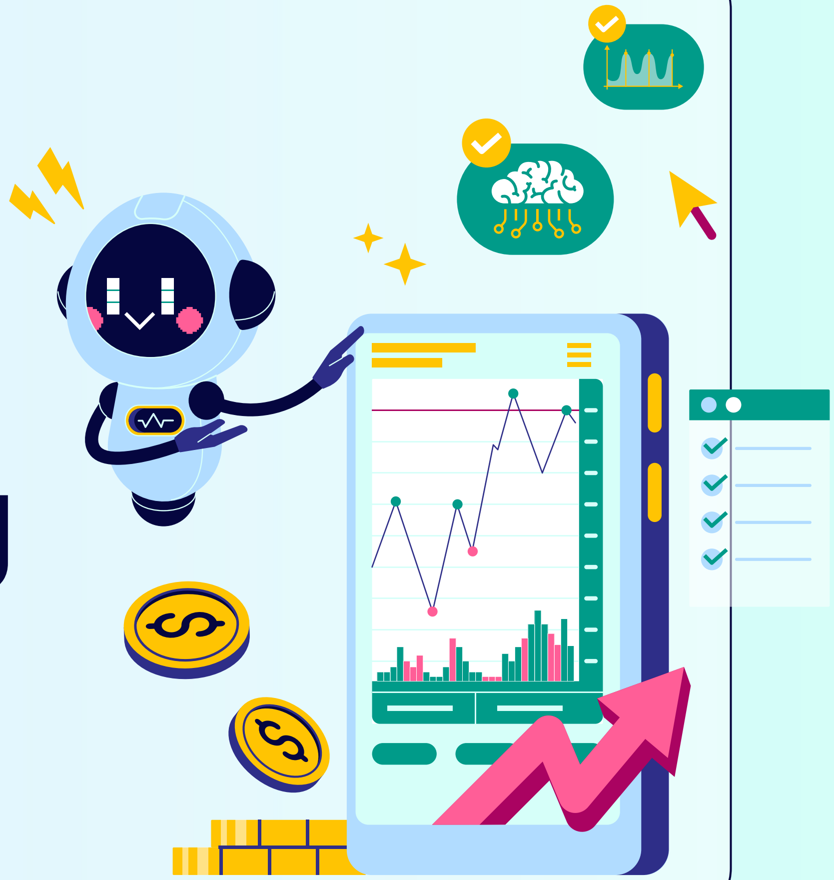
- SVM (too slow using SMOTE)

Oversampling

- Logistic Regression
- Neural Network

03

Model Training



Logistic Regression

```
from sklearn.metrics import *
from sklearn.linear_model import LogisticRegression

logReg = LogisticRegression(penalty="l1", solver='saga', C=.1)
logReg.fit(X_train, y_train)
```

13 Different Configurations

- C value
- transformation
- regularization

Trail #	C (lambda)	transformation	regularization
1	0.1	non	l2
2	1	non	l2
3	0.5	non	l2
4	1.5	non	l2
5	2	non	l2
6	0.1	non	l1
7	1	non	l1
8	0	non	non
9	0.1	poly, power=2	l2
10	1	poly, power=2	l2
11	0.1	poly, power=2	l1
12	1	poly, power=2	l1
13	0	poly, power=2	non

SVM

12 Different Configurations

- C value
- Kernel (transformation)

```
from sklearn.metrics import *
from sklearn.svm import SVC

svm = SVC(kernel='poly', C=.1 , random_state=42)
svm.fit(X_train, y_train)
```

Trail #	C (lambda)	kernel (trans)
1	1	rbf
2	10	rbf
3	0.1	rbf
4	1	linear
5	10	linear
6	0.1	linear
7	1	poly
8	10	poly
9	0.1	poly
10	0.5	poly
11	5	poly
12	50	poly

Neural Network

12 Different Configurations

- C value
- transformation
- regularization

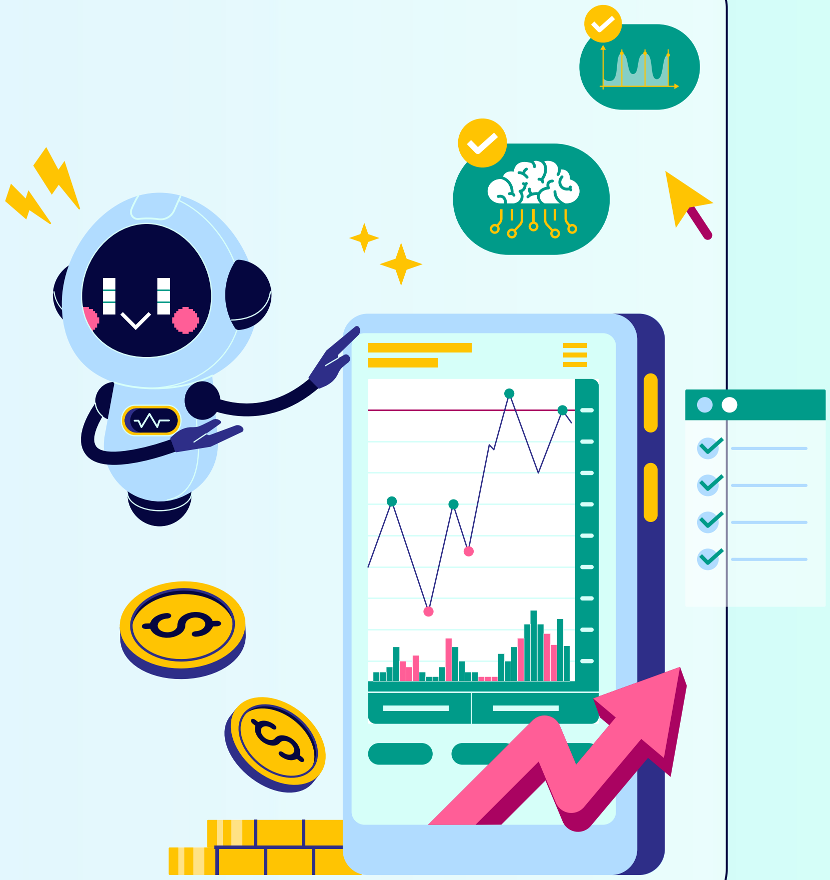
```
from tensorflow.keras import regularizers
import tensorflow as tf
from tensorflow.keras import layers, models

model = models.Sequential([
    layers.Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    layers.Dense(32, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])
```

Trail #	C (lambda)	transformation	regularization
1	0	non	non
2	0.001	non	l2
3	0.1	non	l2
4	0.005		l2
5	0.5		l2
6	0.001	non	l1
7	0.01 (0.1 caused err)	non	l1
8	0	poly, 2 dgr	non
9	0.001	poly, 2 dgr	l2
10	0.1	poly, 2 dgr	l2
11	0.001	poly, 2 dgr	l1
12	0.01 (0.1 caused err)	poly, 2 dgr	l1

04

Result



Logistic Regression

	Best Accuracy
	Best Recall

Trail #	C (lambda)	transformation	regularization	train accuracy	train precision	train recall	train loss	val accuracy	val precision	val recall	val loss
1	0.1	non	l2	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.333
2	1	non	l2	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
3	0.5	non	l2	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
4	1.5	non	l2	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
5	2	non	l2	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
6	0.1	non	l1	0.846	0.921	0.758	5.535	0.935	0.045	0.778	2.331
7	1	non	l1	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
8	0	non	non	0.846	0.921	0.758	5.536	0.935	0.045	0.778	2.334
9	0.1	poly, power=2	l2	0.852	0.963	0.732	5.338	0.971	0.096	0.763	1.043
10	1	poly, power=2	l2	0.852	0.963	0.732	5.339	0.971	0.096	0.763	1.043
11	0.1	poly, power=2	l1	0.852	0.963	0.732	5.338	0.971	0.096	0.763	1.043
12	1	poly, power=2	l1	0.852	0.963	0.732	5.339	0.971	0.096	0.763	1.043
13	0	poly, power=2	non	0.852	0.963	0.732	5.339	0.971	0.096	0.763	1.043

SVM

	Best Accuracy
	Best Recall

Trail #	C (lambda)	kernel (trans)	train accuracy	train precision	train recall	train loss	val accuracy	val precision	val recall	val loss
1	1	rbf	0.865	0.973	0.750	4.876	0.979	0.119	0.708	0.760
2	10	rbf	0.877	0.983	0.877	4.421	0.970	0.086	0.721	1.081
3	0.1	rbf	0.857	0.955	0.750	5.144	0.968	0.080	0.705	1.145
4	1	linear	0.850	0.932	0.756	5.389	0.950	0.053	0.721	1.789
5	10	linear	0.851	0.932	0.757	5.377	0.950	0.053	0.721	1.789
6	0.1	linear	0.850	0.932	0.756	5.389	0.951	0.054	0.721	1.775
7	1	poly	0.850	0.952	0.737	5.424	0.958	0.062	0.705	1.503
8	10	poly	0.864	0.951	0.768	4.899	0.948	0.051	0.726	1.890
9	0.1	poly	0.753	0.967	0.525	8.888	0.983	0.118	0.521	0.597
10	0.5	poly	0.830	0.953	0.694	6.124	0.966	0.071	0.666	1.234
11	5	poly	0.864	0.953	0.766	4.899	0.949	0.052	0.721	1.839
12	50	poly	0.864	0.951	0.767	4.911	0.946	0.050	0.737	1.948

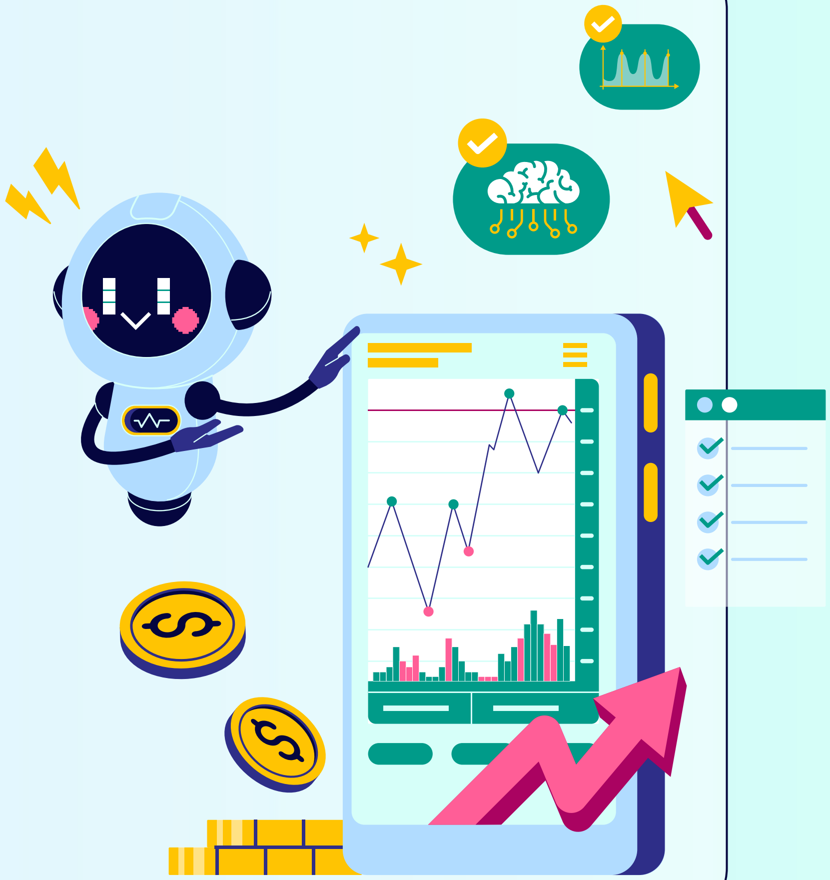
Neural Network

	Best Accuracy
	Best Recall

Trail #	C (lambda)	transformation	regularization	train accuracy	train precision	train recall	train loss	val accuracy	val precision	val recall	val loss
1	0	non	non	0.978	0.987	0.970	0.055	0.986	0.193	0.826	0.045
2	0.001	non	l2	0.970	0.966	0.973	0.116	0.964	0.090	0.901	0.129
3	0.1	non	l2	0.855	0.953	0.747	0.373	0.963	0.076	0.763	0.273
4	0.005		l2	0.947	0.962	0.931	0.178	0.963	0.087	0.895	0.169
5	0.5		l2	0.837	0.898	0.761	0.437	0.914	0.035	0.785	0.432
6	0.001	non	l1	0.941	0.937	0.946	0.188	0.935	0.054	0.956	0.219
7	0.01 (0.1 caused err)	non	l1	0.854	0.941	0.756	0.391	0.952	0.060	0.769	0.370
8	0	poly, 2 dgr	non	0.993	0.989	0.996	0.024	0.987	0.200	0.774	0.052
9	0.001	poly, 2 dgr	l2	0.977	0.967	0.986	0.097	0.966	0.091	0.870	0.122
10	0.1	poly, 2 dgr	l2	0.888	0.939	0.831	0.303	0.946	0.056	0.809	0.258
11	0.001	poly, 2 dgr	l1	0.961	0.968	0.955	0.156	0.967	0.095	0.870	0.144
12	0.01 (0.1 caused err)	poly, 2 dgr	l1	0.857	0.856	0.859	0.329	0.856	0.023	0.846	0.360

05

Conclusion



Analysis

- SVM generally has high accuracy but low validation recall
- Logistic regression has higher recall
- Trail 4 and Trail 6 of neural network has the highest validation recall (0.956) and validation accuracy (0.987) overall, respectively.

In reality, we care more about recall than accuracy

Recall = # all correctly predicted fraud/ # all actual fraud
(False alarm doesn't matter!)

Future Work

As trainers:

- Try more different combinations of regularization, transformation, layers (nn) and so on
- Try to use different models (random forests, decision trees, etc.)
- Use a better computer 🤣

As dataset makers:

- Try the best to make sure no null values
- Try the best to balance the data!

Thanks!

Do you have any questions?