TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Credit Fraud Detection

Group 17

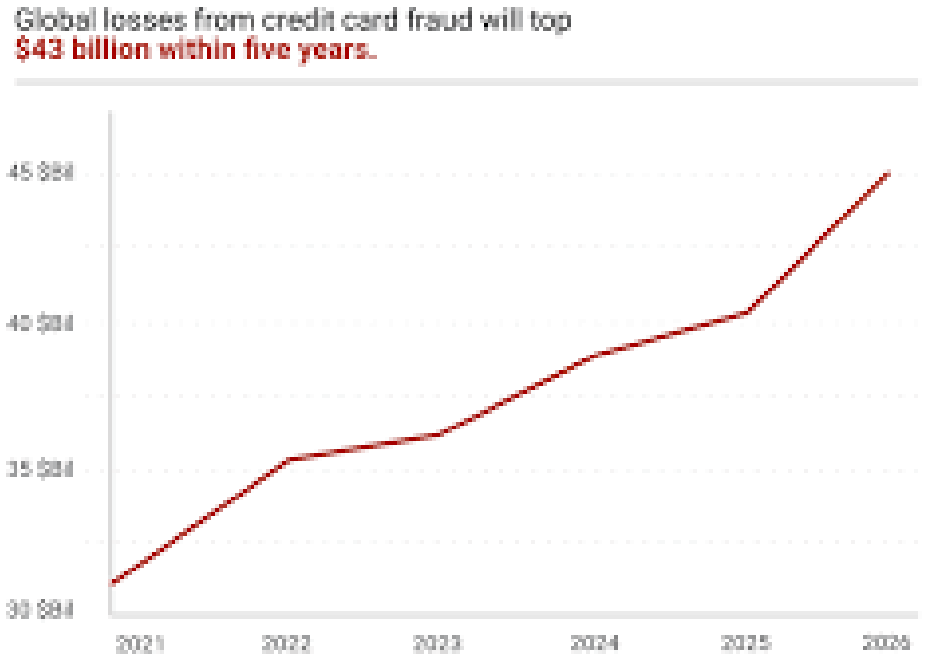
Chu Trung Anh – 20225564
Vũ Hoàng Nhật Anh – 20225471
Trần Nhật Minh – 20225511
Trần Nam Tuấn Vượng - 20225540

HUST

Table of contents

I. Introduction

- Credit fraud (Credit card fraud) is a term for illegal activities commited by criminals using payment cards with or without the authorization of card owners.
- The amount of money lost and number of customers experienced being frauded is increasing over the years and still remains one of crucial problems in finance world.



I. Introduction

- The problem credit card fraud classification is a binary classification problem with aim to detect the fraudulent transactions from the legitimate ones with a good true positive percentage.
- As the large amount of transactions and features of transactions make it impossible for human to calculate
- => This is a suitable job for machine learning

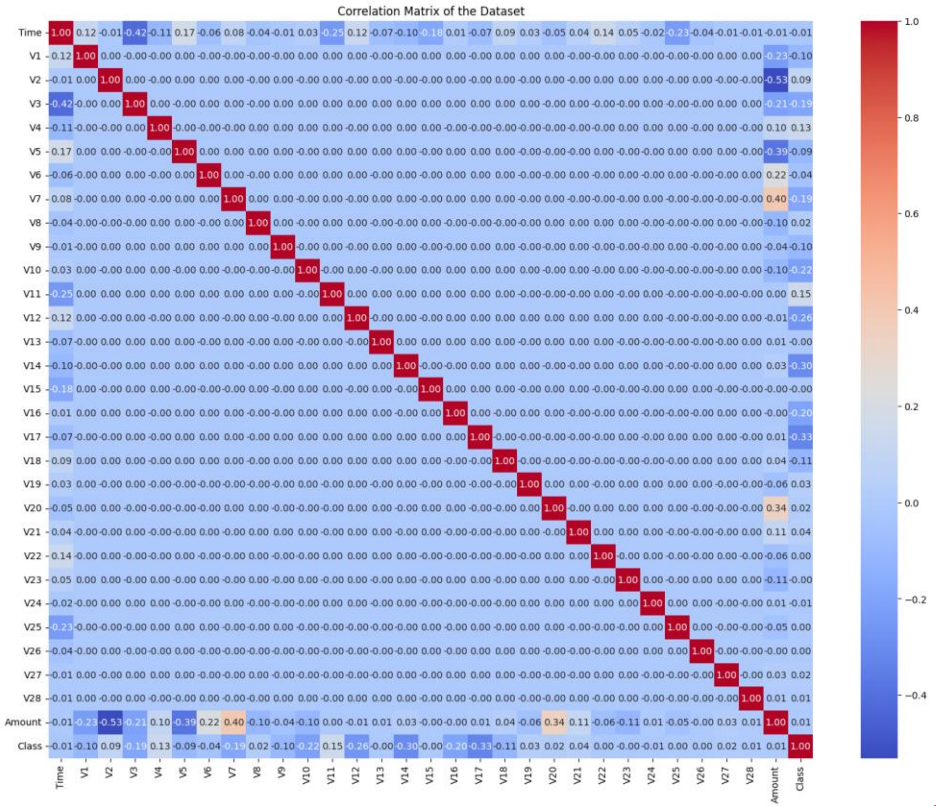
HUST

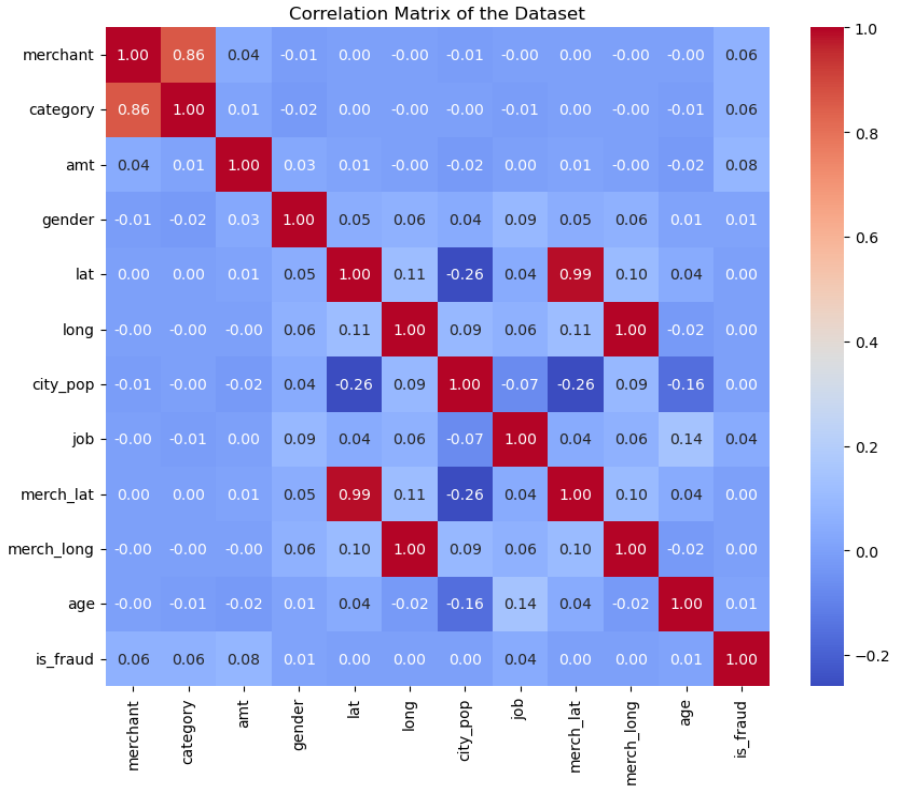
Table of contents

II. Dataset and preprocessing

- We use 2 datasets:

	Dataset1	Dataset3
Number of Fraud	492	9,651
Number of Legit	284,807	1,842,743





II. Dataset and preprocessing

- The data is given in its raw forms:

amt	first	last	gender	street	...	lat	long	city_pop	job	dob	trans_num	unix_time	merch_lat
4.97	Jennifer	Banks	F	561 Perry Cove	...	36.0788	-81.1781	3495	Psychologist, counselling	1988-03-09	0b242abb623afc578575680df30655b9	1325376018	36.011293
107.23	Stephanie	Gill	F	43039 Riley Greens Suite 393	...	48.8878	-118.2105	149	Special educational needs teacher	1978-06-21	1f76529f8574734946361c461b024d99	1325376044	49.159047
220.11	Edward	Sanchez	M	594 White Dale Suite 530	...	42.1808	-112.2620	4154	Nature conservation officer	1962-01-19	a1a22d70485983eac12b5b88dad1cf95	1325376051	43.150704
45.00	Jeremy	White	M	9443 Cynthia Court Apt. 038	...	46.2306	-112.1138	1939	Patent attorney	1967-01-12	6b849c168bdad6f867558c3793159a81	1325376076	47.034331
41.96	Tyler	Garcia	M	408 Bradley Rest	...	38.4207	-79.4629	99	Dance movement psychotherapist	1986-03-28	a41d7549acf90789359a9aa5346dcb46	1325376186	38.674999

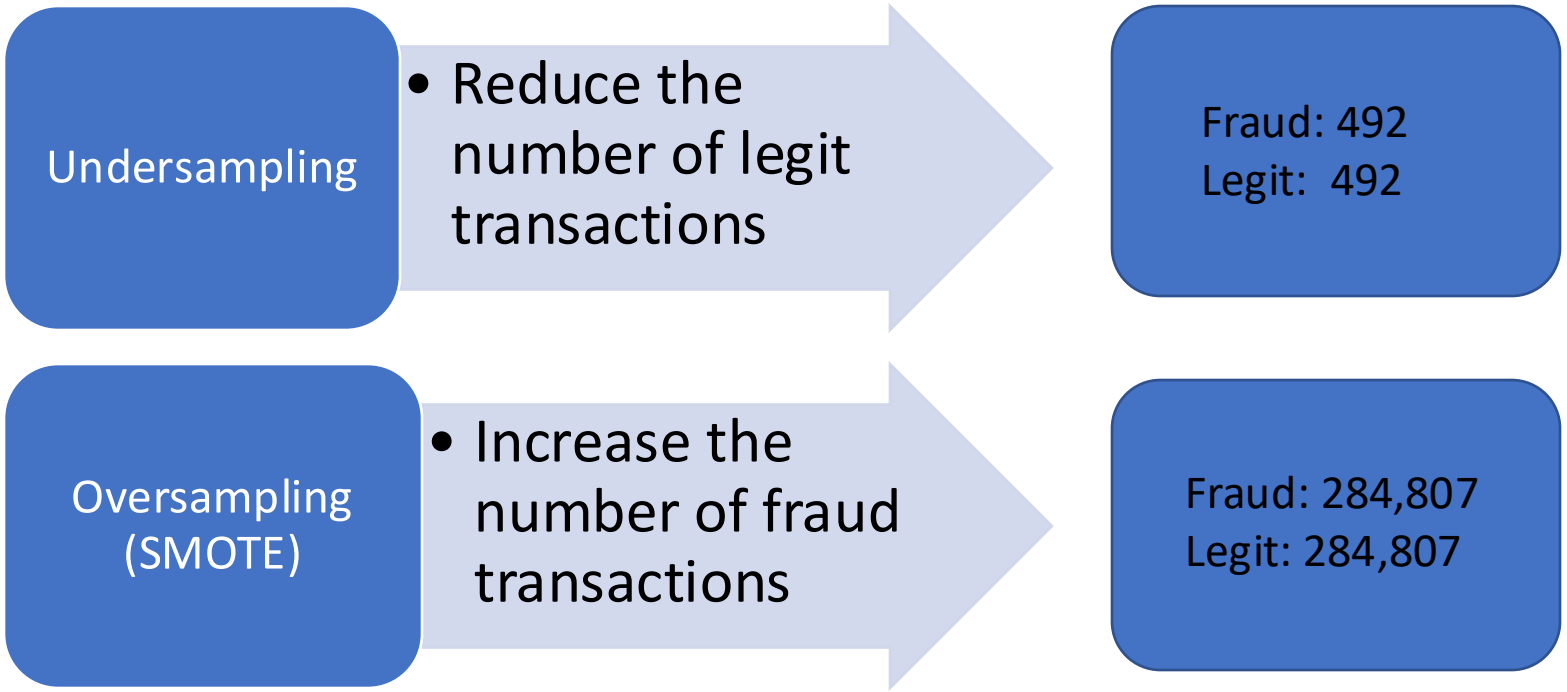
- Normalize the data: Transform the data into numerical values in range (0,1)
  - Remove redundant attributes: names, addresses can be inferred from other attributes…
  - Scale numerical attributes: Min-max scaling $f(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$
  - Encode text label attributes: Target encoding $Weighted\ mean = \frac{n * Category\ mean + m * Overall\ mean}{n + m}$

II. Dataset and preprocessing

merchant	category	amt	gender	lat	long	city_pop	job	merch_lat	merch_long	age
0.013551	0.013038	0.000137	0.0	0.464877	0.753721	0.001194	0.003322	0.462162	0.738663	0.135417
0.009914	0.012645	0.003670	0.0	0.980089	0.112922	0.000043	0.002473	0.991000	0.113337	0.239583
0.001899	0.002177	0.007569	1.0	0.710316	0.215853	0.001421	0.021425	0.749328	0.217713	0.406250
0.002420	0.004106	0.001520	1.0	0.873210	0.218417	0.000659	0.005461	0.905538	0.210678	0.354167
0.003061	0.002819	0.001415	1.0	0.559075	0.783400	0.000026	0.004450	0.569303	0.797770	0.156250
...	...	...	...	...	...	...	...	...	...	...
0.002466	0.001510	0.001477	1.0	0.642432	0.568344	0.000171	0.003762	0.620460	0.577997	0.375000
0.001830	0.001880	0.003829	1.0	0.181730	0.506935	0.009879	0.005631	0.206738	0.494017	0.031250
0.001828	0.001880	0.002967	0.0	0.871842	0.100961	0.001260	0.003193	0.890415	0.086887	0.218750
0.002629	0.002692	0.000241	1.0	0.808648	0.143397	0.000036	0.003768	0.802415	0.132468	0.385417
0.003033	0.002177	0.001283	1.0	0.448293	0.471640	0.039901	0.002742	0.470158	0.479313	0.093750

II. Dataset and preprocessing

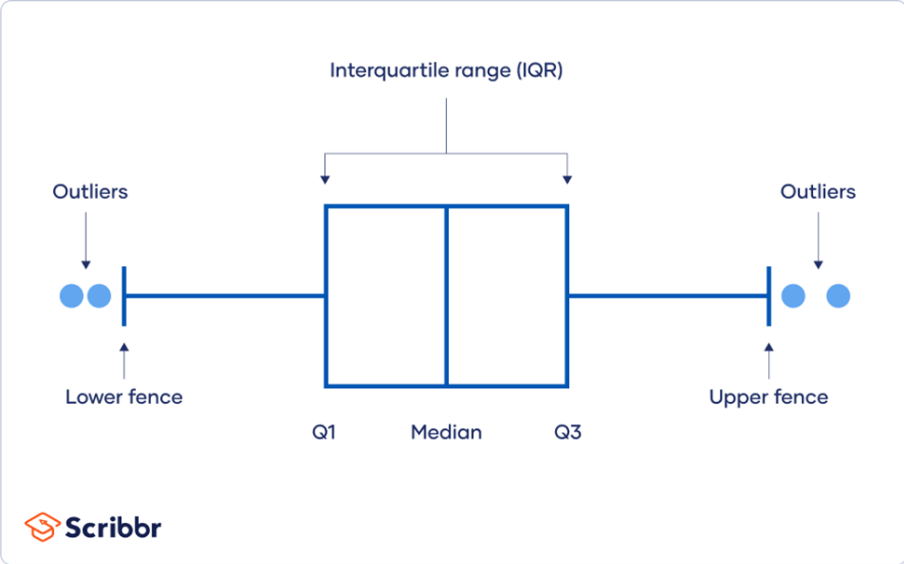
- Techniques to handle imbalance dataset:

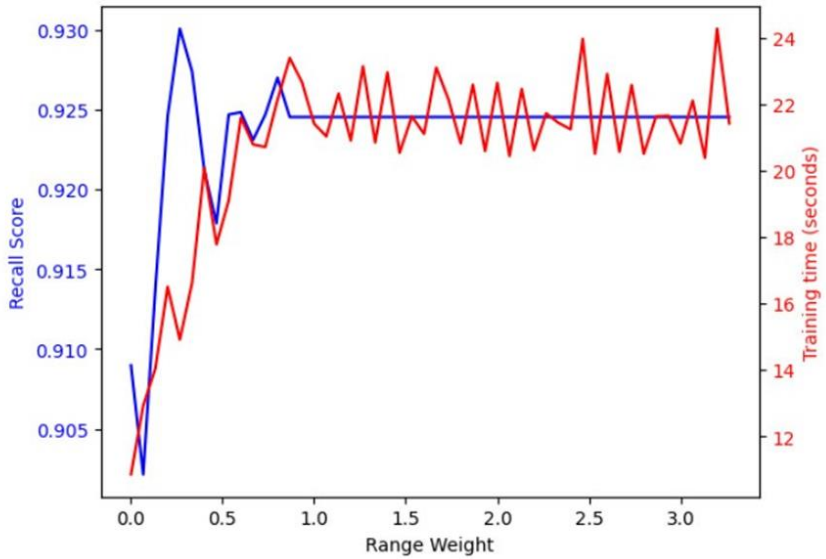


II. Dataset and preprocessing

- Outlier detection using interquartile range method:
  - Identify the first quartile (Q1), the median, and the third quartile (Q3).
  - Calculate IQR = Q3 – Q1.
  - Calculate upper fence = Q3 + (w * IQR) and lower fence = Q1 – (w * IQR)

  where w is the range weight (usually set as 1.5).
  - Use fences to indicate the outliers which are all the values that fall outside the fences.





II. Dataset and preprocessing

- Dimensionality Reduction

Principal Component Analysis

- Standardize the range of continuous initial variables.
- Compute the covariance matrix to identify correlations.
- Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components.
- Create a feature vector to decide which principal components to keep.
- Recast the data along the principal components axes.

II. Dataset and preprocessing

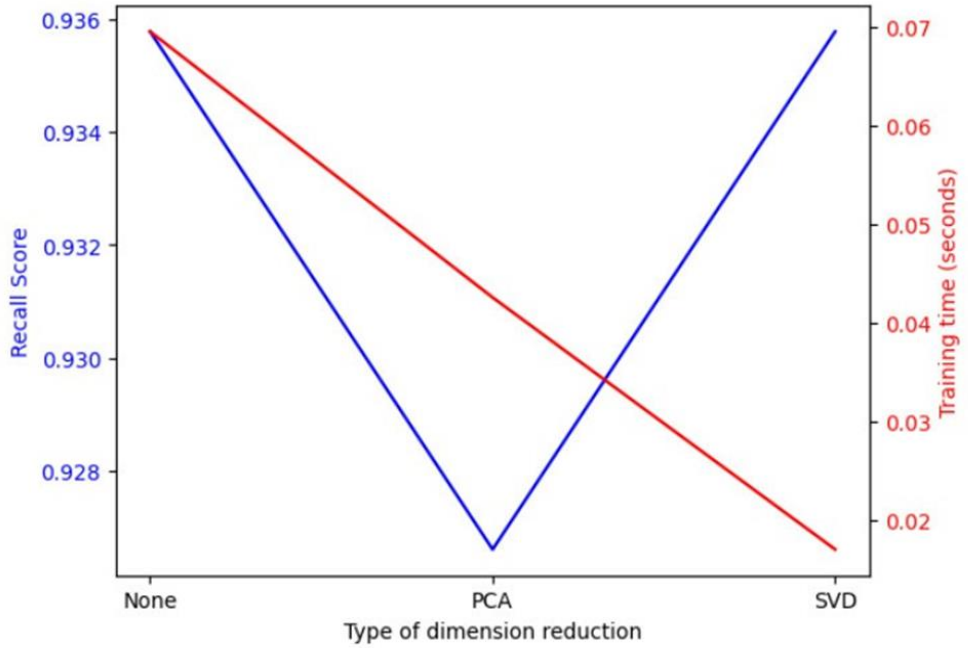
- Dimensionality Reduction

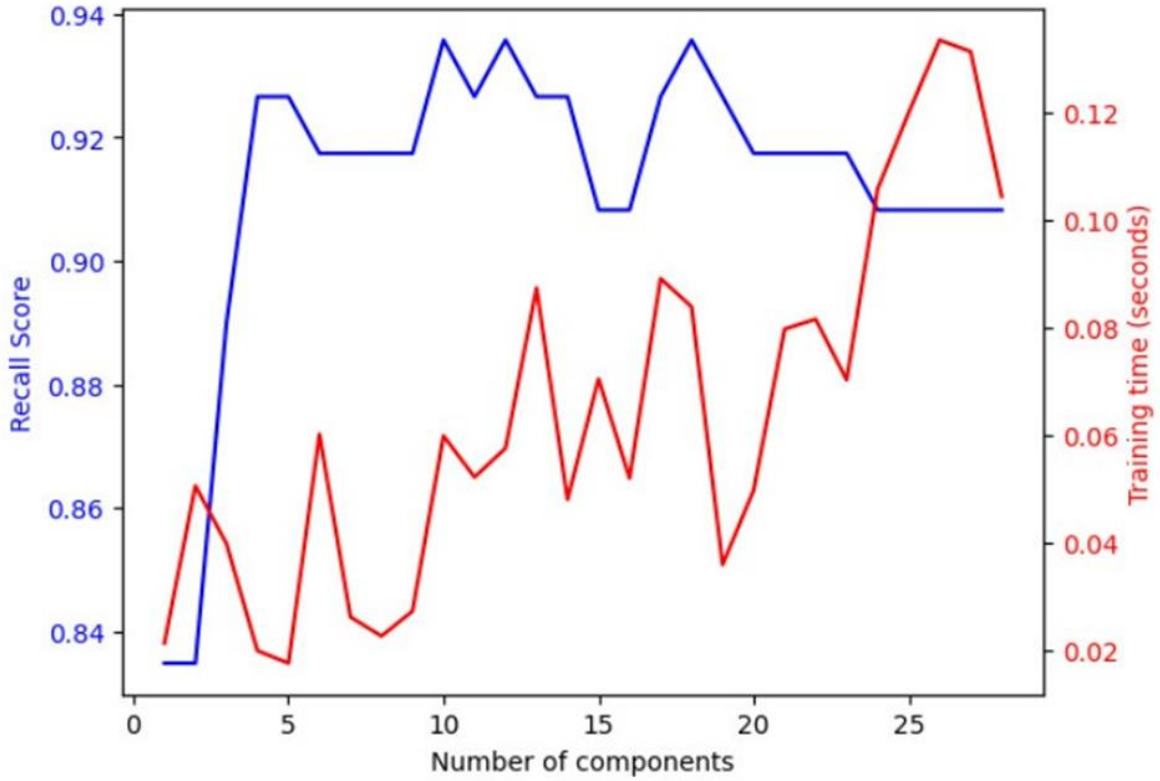
Truncated Singular Value Decomposition

- The SVD of m x n matrix A is given by the formula $A = U\sum V^T$
- U: m x m matrix of the orthonormal eigenvectors of AA^T
- V^T: transpose of a n x n matrix containing the orthonormal eigenvectors of AA^T
- $\sum$: diagonal matrix with r elements equal to the root of the positive eigenvalues of AA^T
- A given mxn matrix truncated SVD will produce matrices with the specified number of columns, whereas a normal SVD procedure will produce with m columns. It means that it will drop off all features except the number of features provided to it.

II. Dataset and preprocessing

- Dimensionality Reduction





HUST

Table of contents

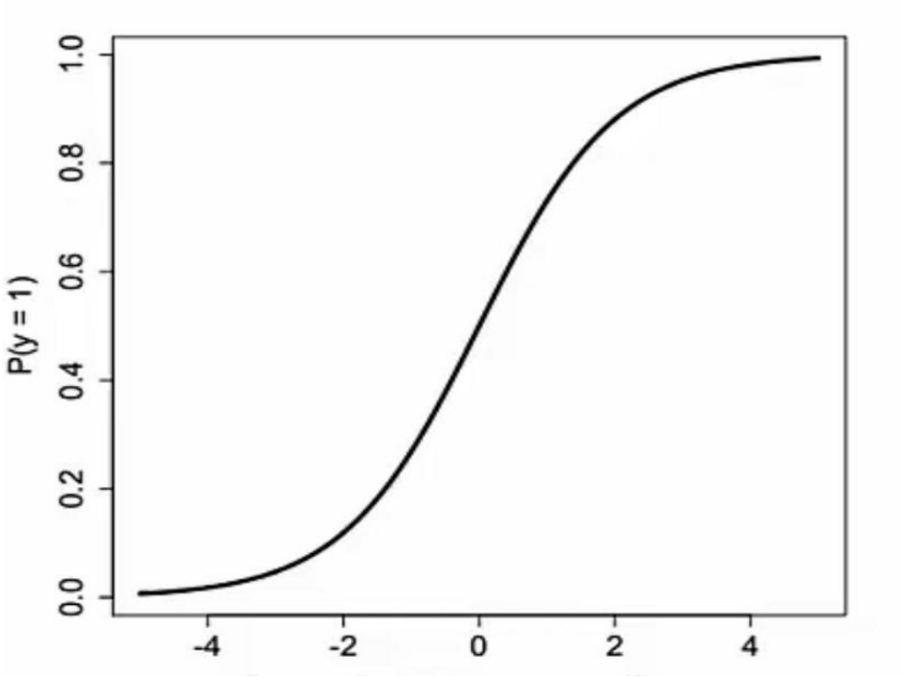
III. Methodology

Libraries used:

- Numpy, Pandas, Matplotlib.
- Sklearn, imblearn.
- Keras.

III. Methodology

- 1. Logistic Regression



$$P_{(y=1)} = \frac{1}{1 + e^{-(w_0 + w_1 x_1 + \cdots + w_n x_n)}}$$

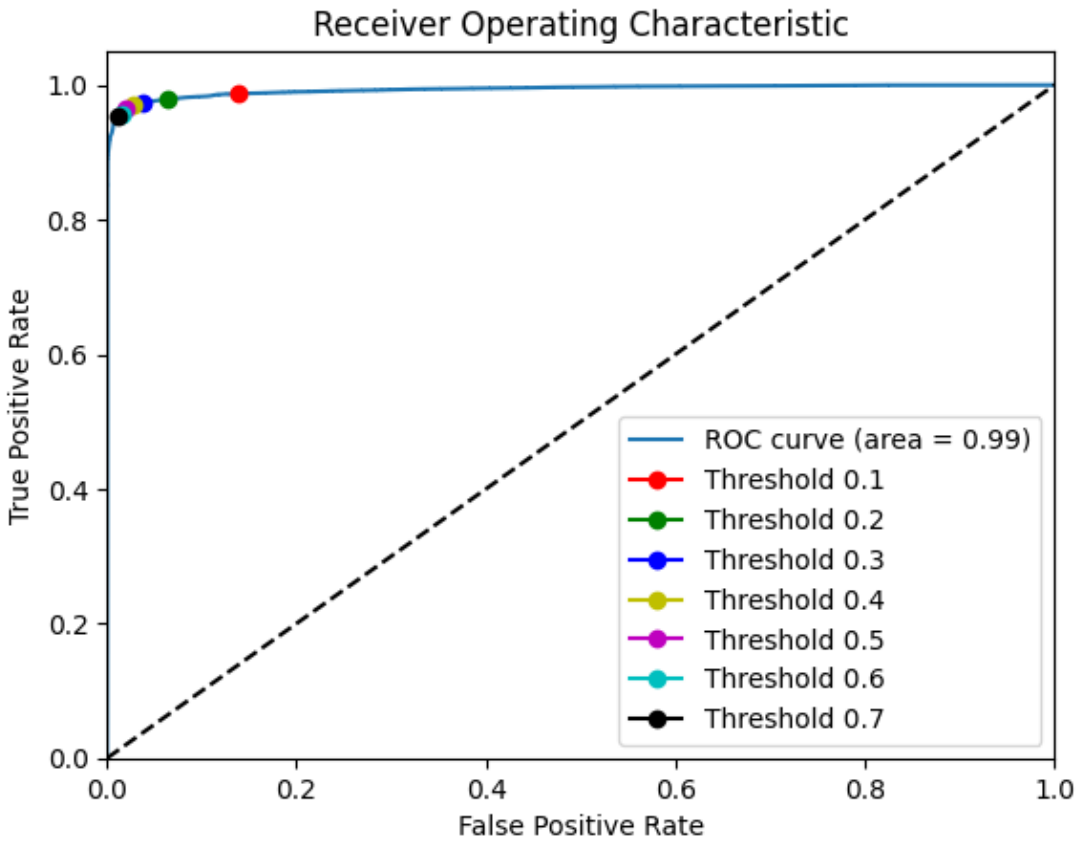
=> Learning weight vector: $w^* = arg \max_w P(y|X, w)$

We use **Cross Entropy** $J(w, x_i, y_i) = \sum_{i=1}^{n} -[y_i \, log\left(f\left(w^T x_i\right)\right) + (1 - y_i) \, log\left(1 - f\left(w^T x_i\right)\right]$

III. Methodology

- 1. Logistic Regression

Hyperparameter Tuning



- ***True Positive Rate*** **(**also known as Sensitivity/Recall) represents the proportion of actual positive that are correctly identified.

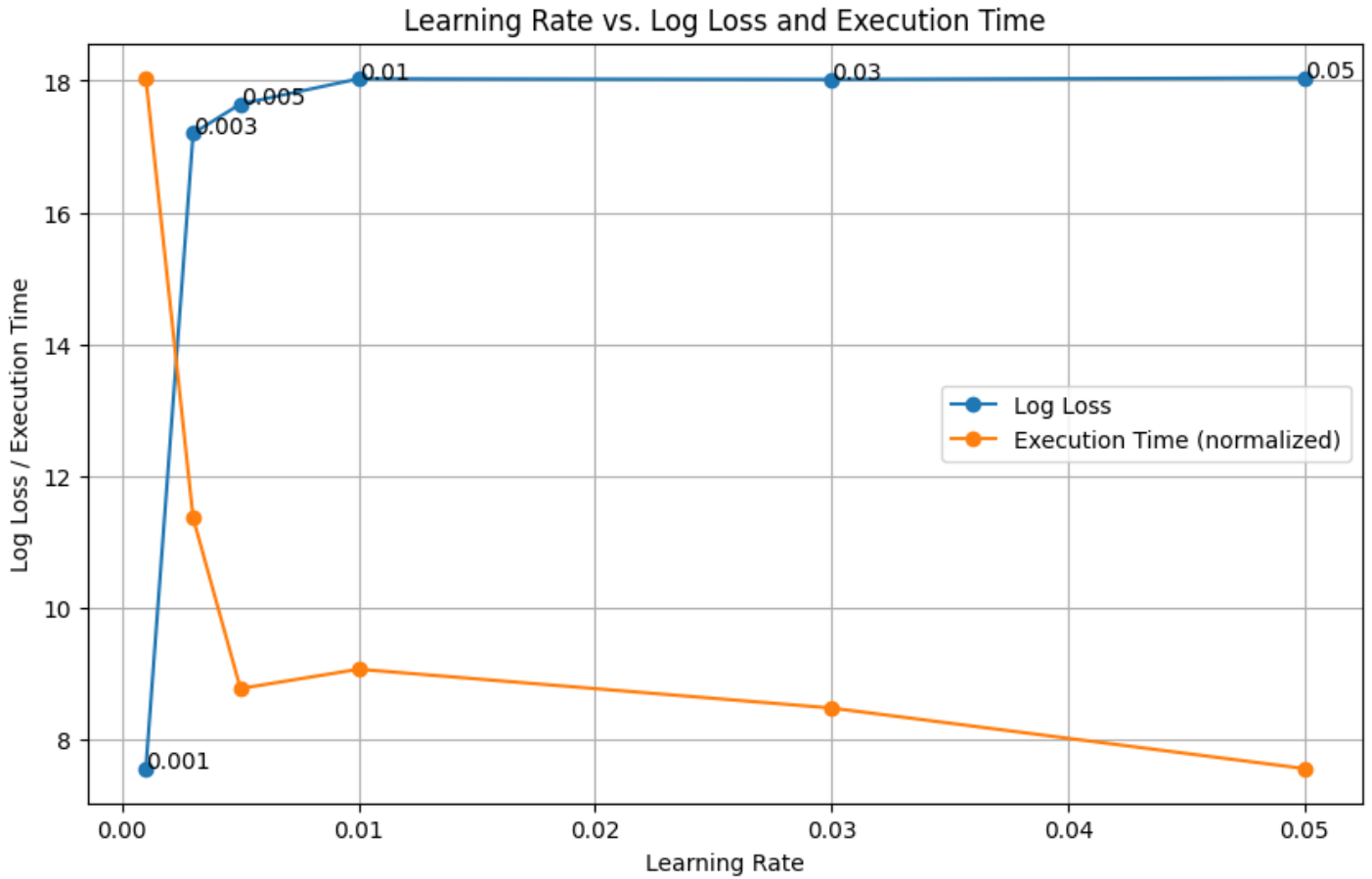
  → *A higher value indicates a better ability of the model to correctly classify positive instances.*

- **False Positive Rate** represents the proportion of negative instances that are incorrectly classified as positive out of all actual negative instances. In simple words, this is the wrong alert rate.

III. Methodology

- 1. Logistic Regression

  Hyperparameter Tuning



III. Methodology

- 1. Logistic Regression

  Model and Result

➢ Self-implement Logistic Model using Gradient Descent algorithm

➢ Build-in Logistic Regression using Gradient Descent algorithm

➢ Default Logistic Regression in Sklearn which use: *'liblinear'*, *'newton-cg'*, *'sag'*, *'saga'*, and *'lbfgs'* algorithm

III. Methodology

- 1. Logistic Regression

Model and Result

```
def sigmoid(self, z):
    # Clip input values to be between -250 and 250
      z = np.clip(z, -500, 500)
      return 1 / (1 + np.exp(-z))

def train(self, X, y):
    num_samples, num_features = X.shape

    # 1. Initialize weights and bias
    self.weights = np.zeros(num_features)
    self.bias = 0

    # 2. Gradient descent loop
    for i in range(self.num_iterations):
        hyperplane = np.dot(X, self.weights) + self.bias
        y_predicted = self.sigmoid(hyperplane)

        # 3. Compute gradients
        self.weights -= self.learning_rate * np.dot(X.T, (y_predicted - y))

def predict(self, X):
    linear_model = np.dot(X, self.weights) + self.bias
    y_predicted = self.sigmoid(linear_model)
    y_predicted_cls = [1 if i >= 0.2 else 0 for i in y_predicted]
    return np.array(y_predicted_cls)
```

$$\Rightarrow S(z) = \frac{1}{1 + e^{-z}}$$

$$\Rightarrow z = w_0 + w_1 x_1 + \cdots + w_n x_n$$

$$\Rightarrow w_{update} = w - \alpha * \underbrace{x_i (y_i - S(z))}_{\nabla J(w, x_i, y_i)}$$

III. Methodology

- 1. Logistic Regression

Model and Result

	Self Implement		SGDClassifier		Default LR		Default LR Tuning	
	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud
Precision	0.50	0.00	1.00	0.50	0.97	0.98	0.98	0.93
Recall	1.0	0.00	0.00	1.00	0.98	0.96	0.93	0.98
F1	0.67	0.00	0.00	0.67	0.97	0.97	0.95	0.96
Accuracy	0.50		0.50		0.97		0.96	

III. Methodology

- 1. Logistic Regression

Model and Result

	Self Implement		SGDClassifier		Default LR		Default LR Tuning	
	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud
Precision	0.50	0.00	1.00	0.50	0.97	0.98	0.98	0.93
Recall	1.0	0.00	0.00	1.00	0.98	0.96	0.93	0.98
F1	0.67	0.00	0.00	0.67	0.97	0.97	0.95	0.96
Accuracy	0.50		0.50		0.97		0.96	

III. Methodology

- 1. Logistic Regression

Model and Result

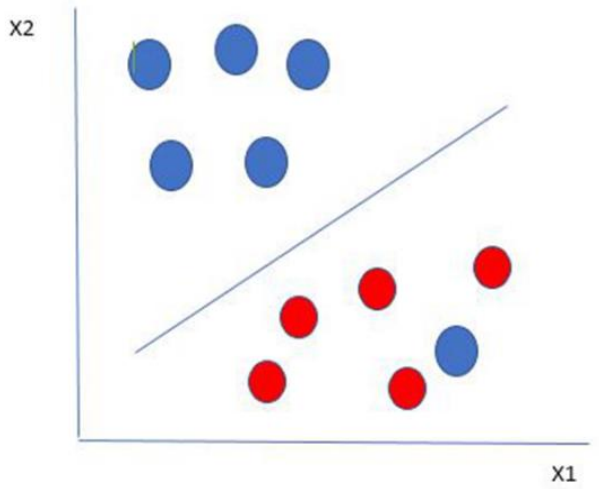
	Default LR		Default LR Tuning	
	N Fraud	Fraud	N Fraud	Fraud
Precision	0.97	0.98	0.98	0.93
Recall	0.98	**0.96**	0.93	**0.98**
F1	0.97	0.97	0.95	0.96
Accuracy	0.97		0.96	

III. Methodology

- 2.SVM

For Linear SVM classifier :

$$\hat{y} = \begin{cases} 1 \; if \; w^T + b \geq 0 \\ 0 \; otherwise \end{cases}$$



Optimization:

- For Hard margin linear SVM classifier:

$$\min \frac{1}{2}||w||^2 \text{ conditioned on } y_i(w^T x_i + b) \geq 1 \text{ for } i = 1,2,3, \ldots, m$$

- For Soft margin linear SVM classifier:

$$\min\left(\frac{1}{2}||w||^2 + C\sum_{i=1}^{m}\zeta_i\right) \text{ conditioned on } y_i(w^T x_i + b) \geq 1 - \zeta_i \text{ for } i = 1,2,3, \ldots, m$$

III. Methodology

- 2.SVM

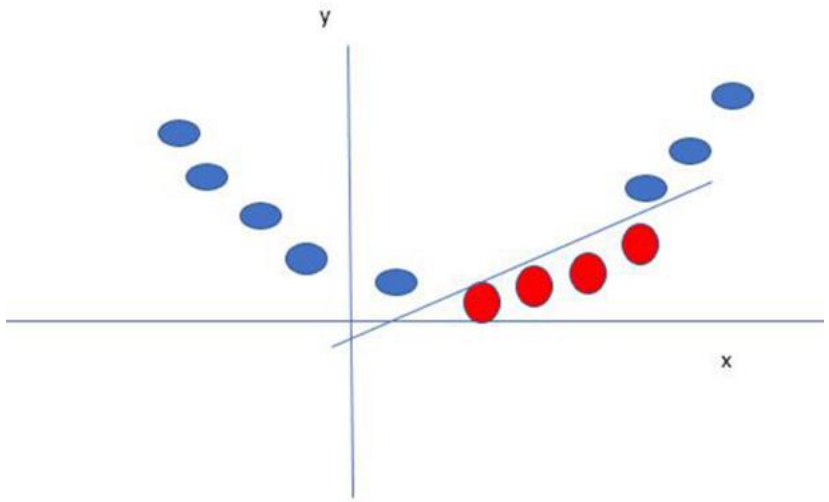
Popular kernel functions in SVM :

$$\text{Linear} : K(w,b) = w^T x + b$$

$$\textit{Polynomial} : K(w,b) = (\gamma w^T + b)^N$$

$$\textit{Gaussian RBF} : K(w,b) = exp(-\gamma ||x_i - x_j||^n)$$

$$\text{Sigmoid} : K(x_i, x_j) = tanh(\alpha {x_i}^T x_j + b)$$



- 2.SVM

Hyperparameter tuning

- Kernel

The learning of the hyperplane in SVM is done by transforming the problem using a parameter called kernel to specify which type of SVM kernel function will be used.

- Regularization

The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example.

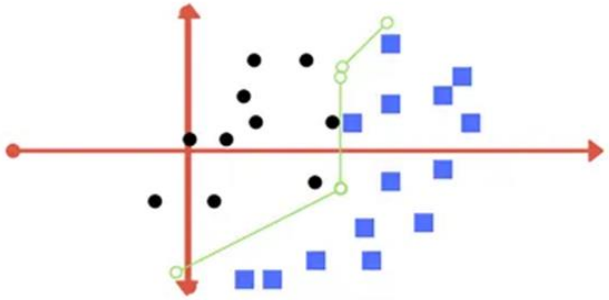
- Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Whereas high gamma means the points close to plausible line are considered in calculation.

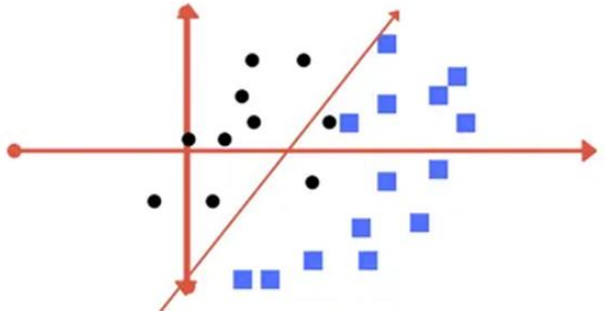
III. Methodology

- 2.SVM

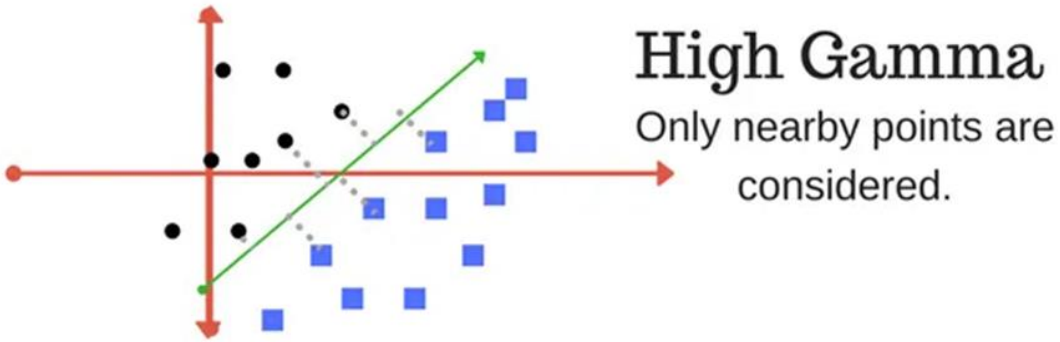
Hyperparameter tuning



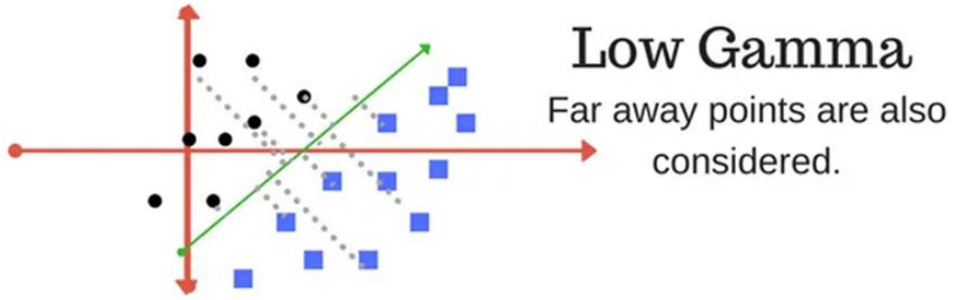
High regularization



Low regularization



High gamma

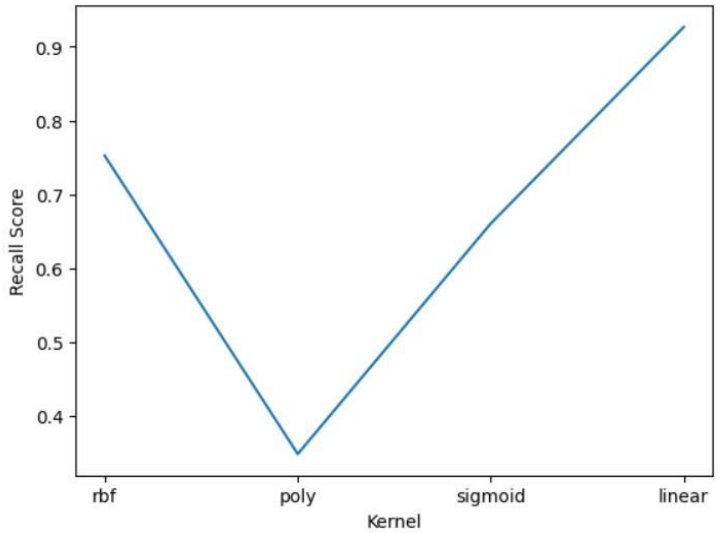


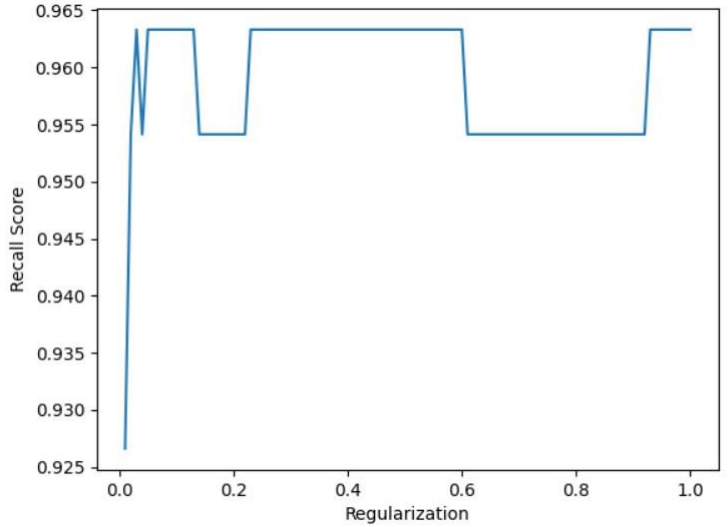
Low regularization

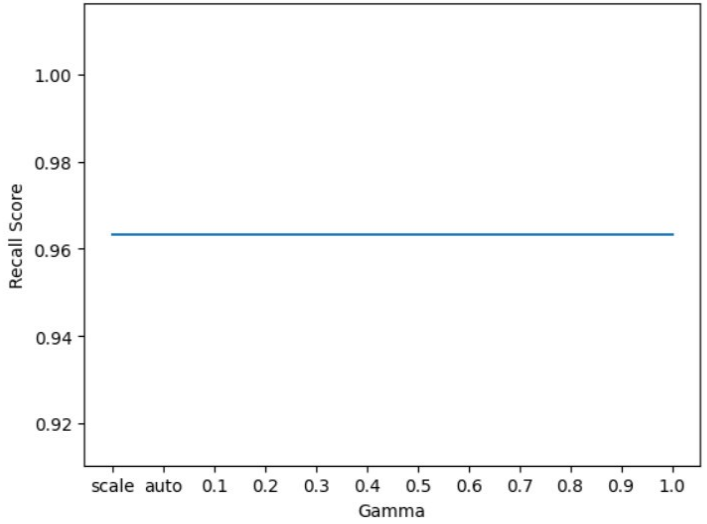
III. Methodology

- 2.SVM

Hyperparameter tuning







III. Methodology

- 2.SVM

Result

- Undersampling:

	precision	recall	f1-score	support
Not fraud	0.92	0.98	0.95	88
Fraud	0.98	0.92	0.95	92
accuracy			0.95	180
macro avg	0.95	0.95	0.95	180
weighted avg	0.95	0.95	0.95	180

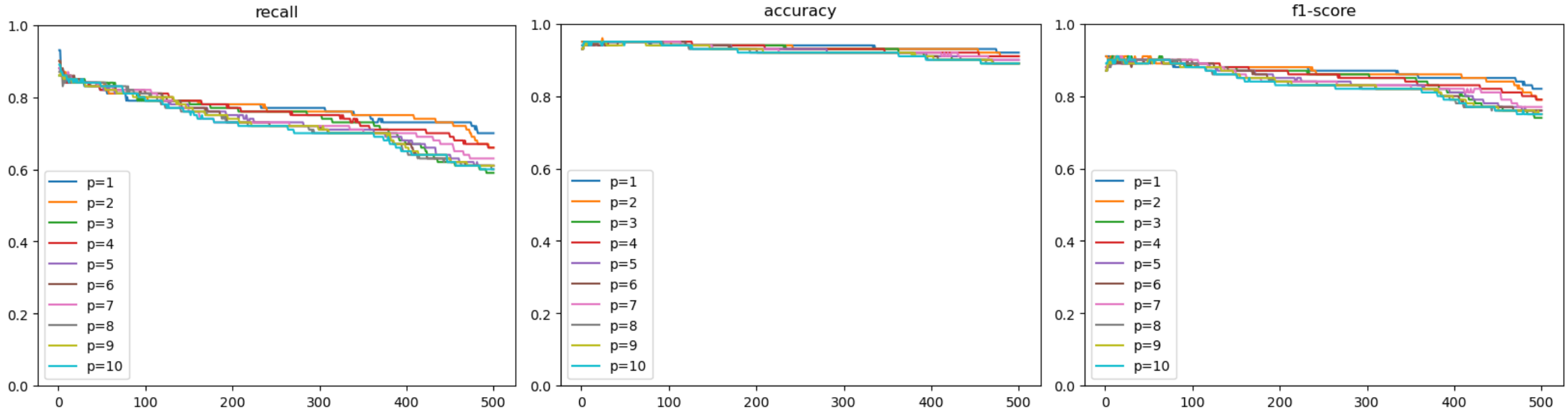
- Oversampling:

	precision	recall	f1-score	support
Not fraud	0.91	0.99	0.95	272236
Fraud	0.99	0.89	0.94	241788
accuracy			0.95	514024
macro avg	0.95	0.94	0.94	514024
weighted avg	0.95	0.95	0.95	514024

III. Methodology

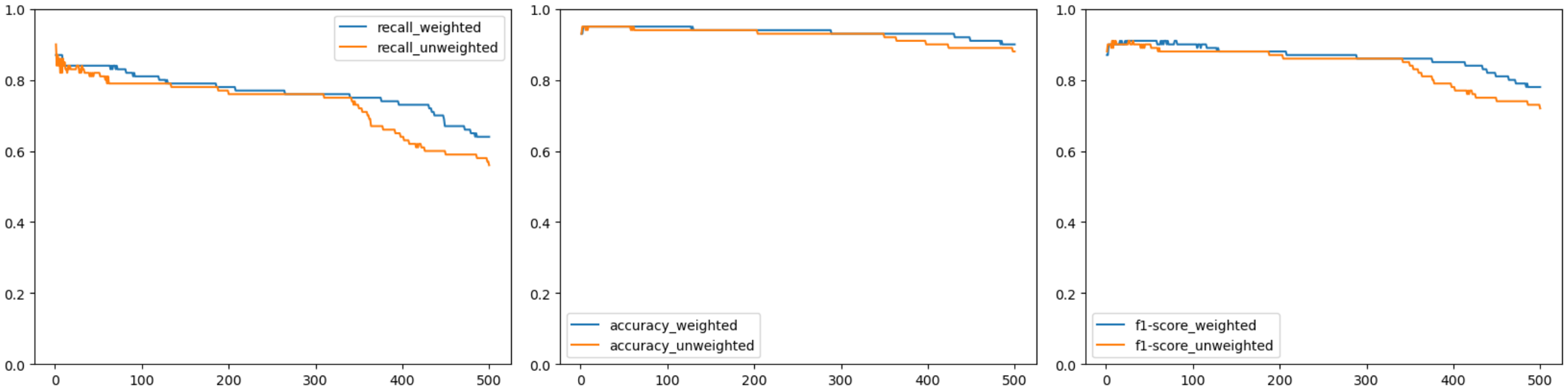
3.KNN

- Distance functions:
  - Euclidean Distance: $d(x,y) = \sqrt{\sum_{j=1}^{d}(x_j - y_j)^2}$
  - Manhattan Distance: $d(x,y) = \sum_{i=1}^{n}|x_i - y_i|$
  - Minkowski Distance: $d(x,y) = (\sum_{i=1}^{n}|x_i - y_i|^p)^{\frac{1}{p}}$



3.KNN

- Choosing the value of k:
  - Experiment with many values of k and choose the best one.
  - Weight the neighbors: $w = \frac{1}{d(x,y)}$



III. Methodology

- 4 . Decision Tree

Information Gain

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \varepsilon Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Gini Impurity

$$G(S) = 1 - \sum_{i=1}^{c} (p_i)^2$$

III. Methodology

- 4 . Decision Tree

Information Gain

$$Entropy(S) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \varepsilon Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

⇩

ID3

Gini Impurity

$$G(S) = 1 - \sum_{i=1}^{c} (p_i)^2$$

⇩

CART

III. Methodology

- 4 .Decision Tree

Hyperparameter Tuning

```
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [10, 15,20],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 5],
    'max_features': ['sqrt', 'log2']
}
```

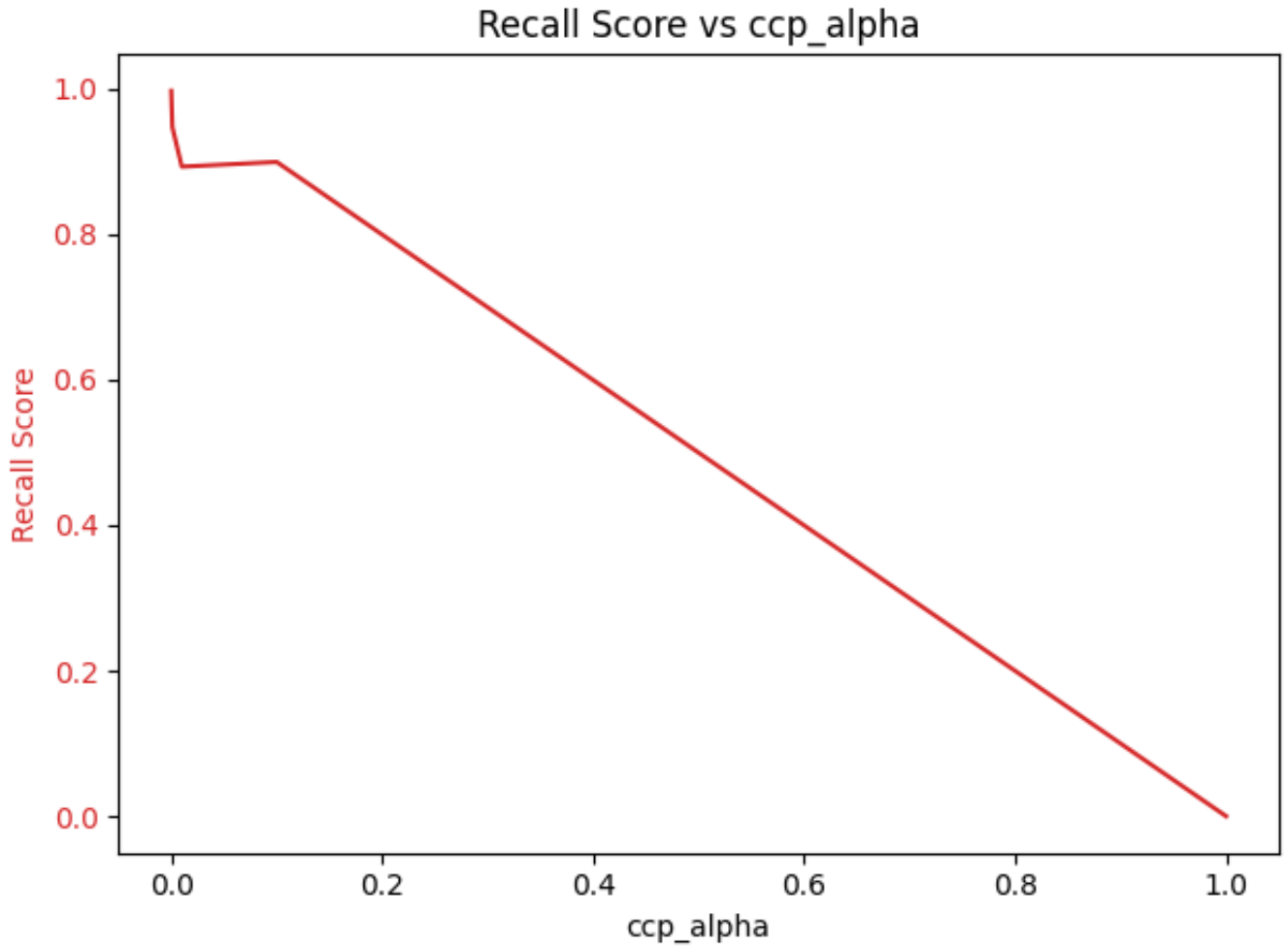
⇩ GridSearchCV on Recall score

```
Best parameters:  {'criterion': 'gini', 'max_depth': 20, 'max_features': 'sqrt', 'min_samples_leaf': 1, 'min_samples_split': 2}
Best score:  0.9982457837257971
```

III. Methodology

- 4 . Decision Tree

Hyperparameter Tuning

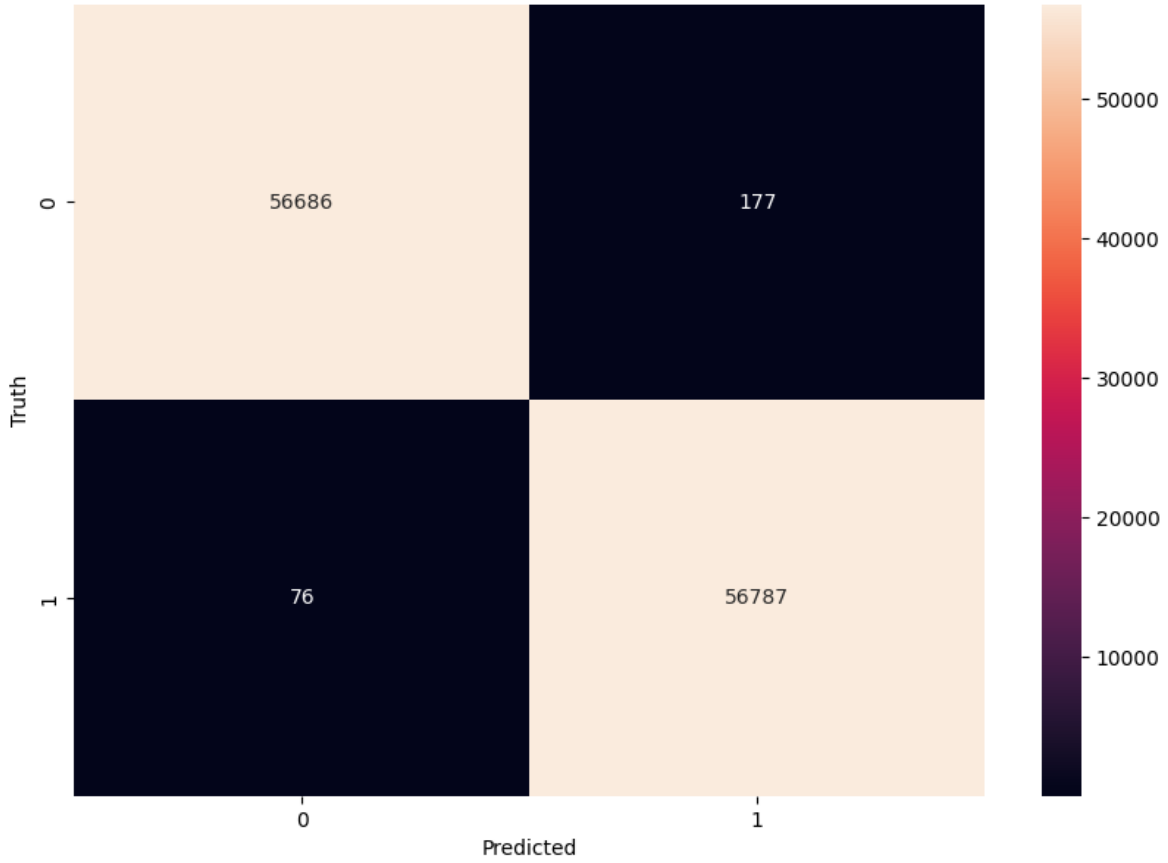


III. Methodology

- 4 . Decision Tree

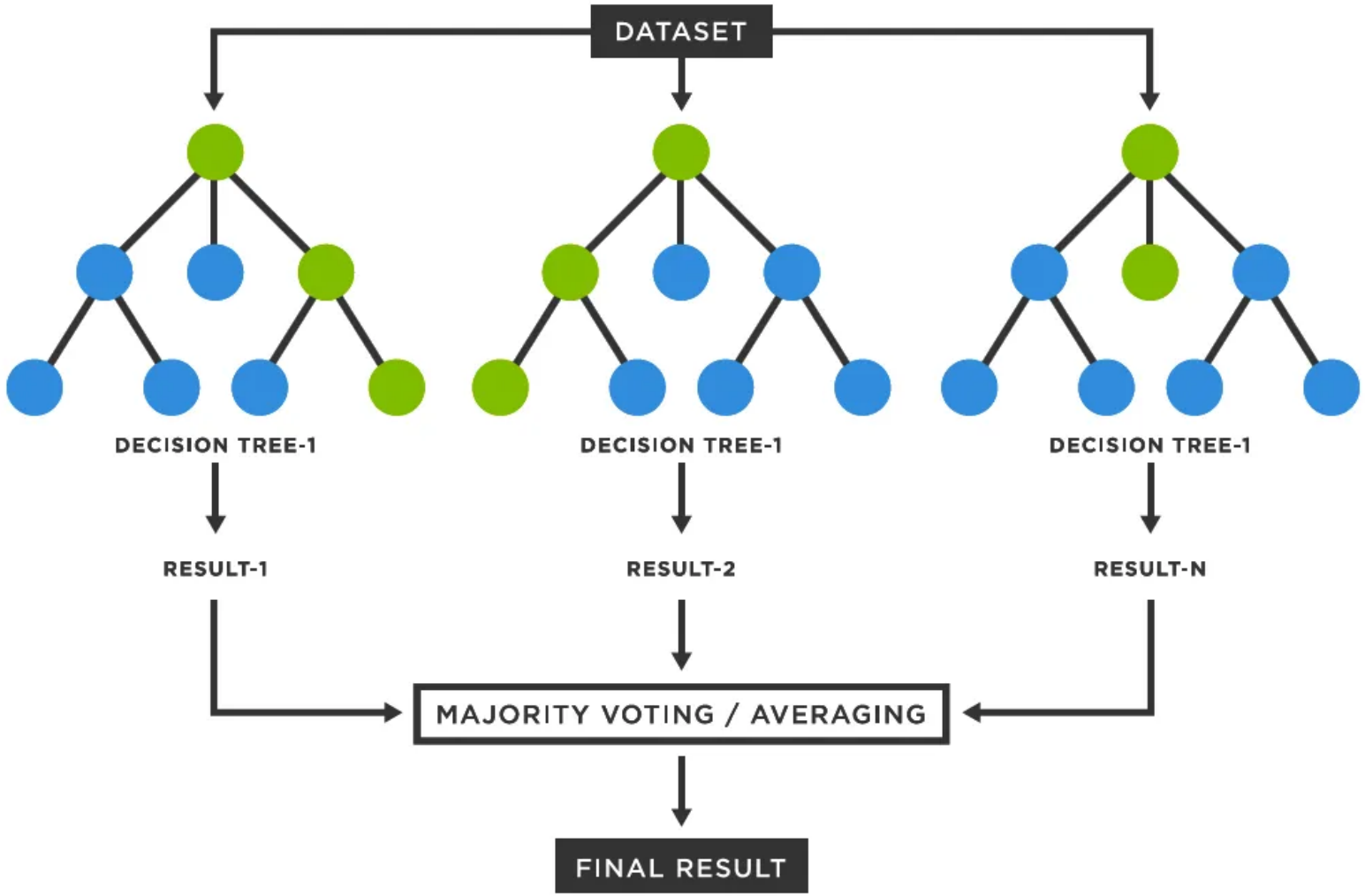
Model and Result

	Decision Tree	
	N Fraud	Fraud
Precision	1.0	1.0
Recall	1.0	1.0
F1	1.0	1.0
Accuracy	1.0	



III. Methodology

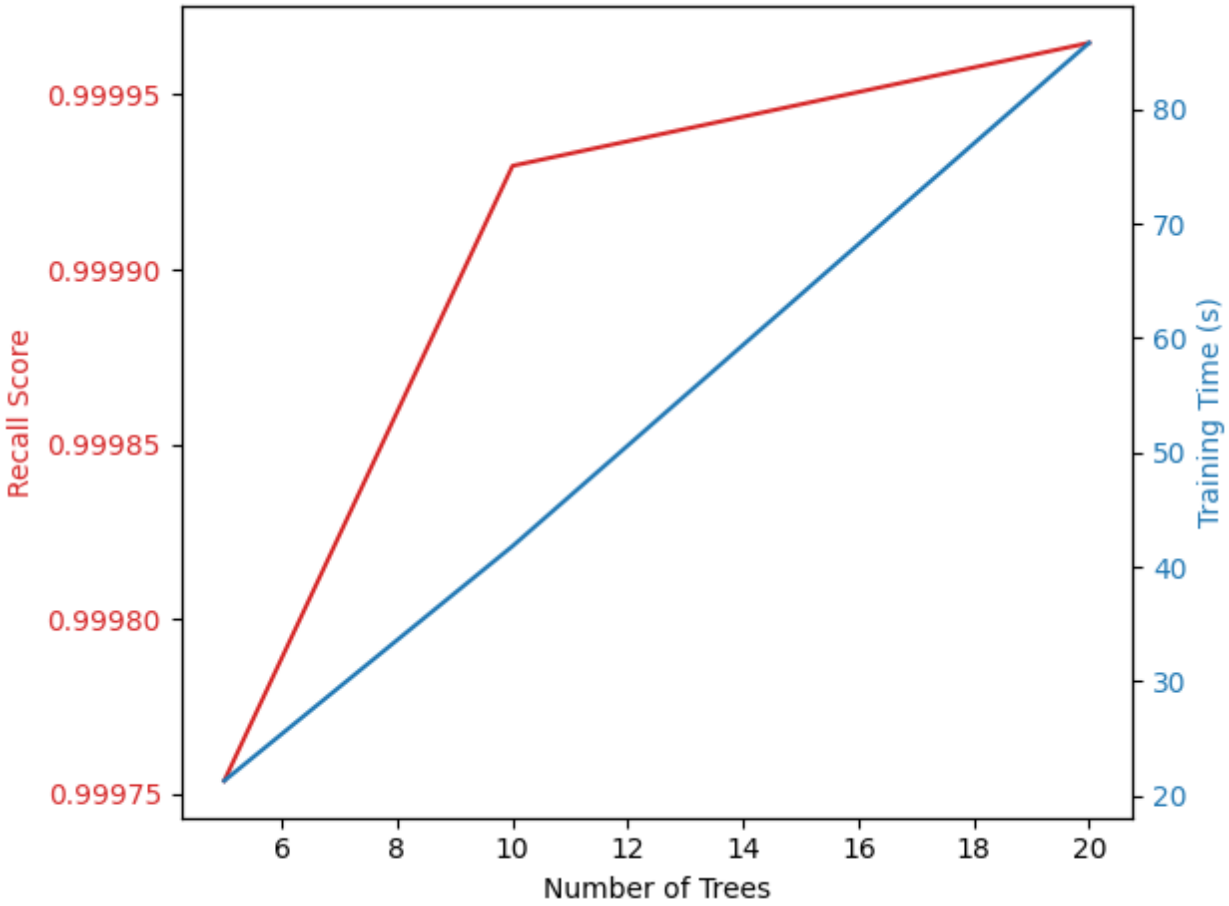
- 5 . Randoms Forests



III. Methodology

- 5 . Randoms Forests

Hyperparameter Tuning



III. Methodology

- 5 . Randoms Forests

Model and Result

- The first model will be based on the theoretical concepts discussed earlier to help us understand how it works, which we will manually perform bootstrap sampling, build decision trees, and aggregate their predictions.

- The second model will use the default Random Forest implementation provided by scikit-learn.

III. Methodology

- 5 . Randoms Forests

Model and Result

1. Bootstrap

```
bootstrap_samples = []
for i in range(3):
    bootstrap_sample = df.sample(n=len(df), replace=True)
    bootstrap_samples.append(bootstrap_sample)
```

2. Individual trees with random subspace

```
from sklearn.tree import DecisionTreeClassifier
for sample in bootstrap_samples:
     dt_model = DecisionTreeClassifier(max_features='sqrt') # Random subspace
     # Extract X,y from sample ...
     dt_model.fit(X, y)
     dt_models.append(dt_model)
```

3. Majority Voting

```
for j in range(len(predictions[0])): #take length of the first tree prediction list
  n_ones = predictions[0][j] +  predictions[1][j]   + predictions[2][j]
  if n_ones > 1:
      final_result = 1
  else:
      final_result = 0
```

- 5 . Randoms Forests

Model and Result

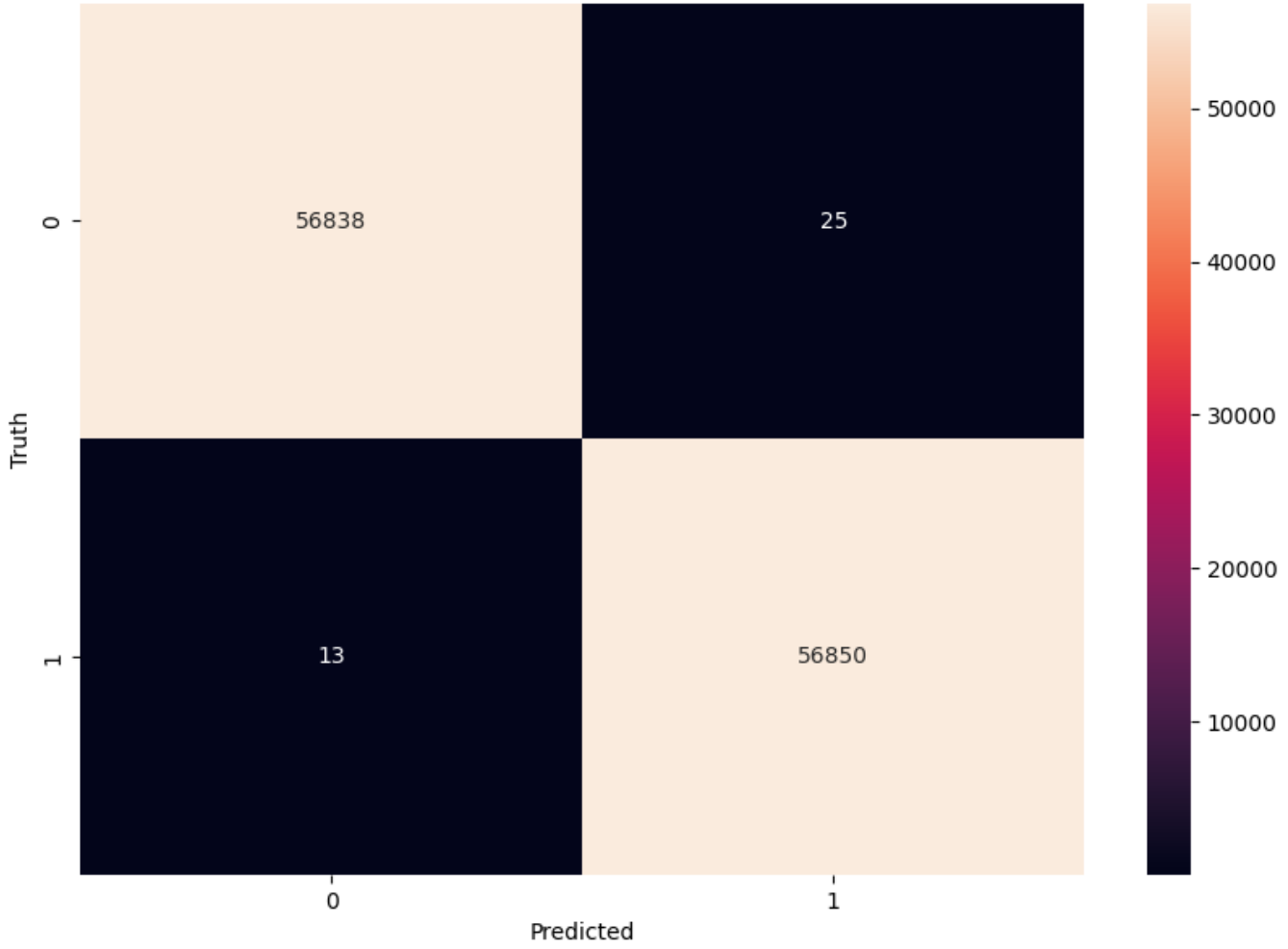
	Tree 1		Tree 2		Tree 3		Majority Voting	
	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud	N Fraud	Fraud
Precision	0.8	1.00	0.84	1.00	0.80	1.00	0.84	1.00
Recall	1.0	0.76	1.0	0.81	1.00	0.75	1.00	0.80
F1	0.89	0.86	0.92	0.90	0.89	0.85	0.91	0.89
Accuracy	0.88		0.91		0.87		0.90	

Measurement on raw imbalance dataset

III. Methodology

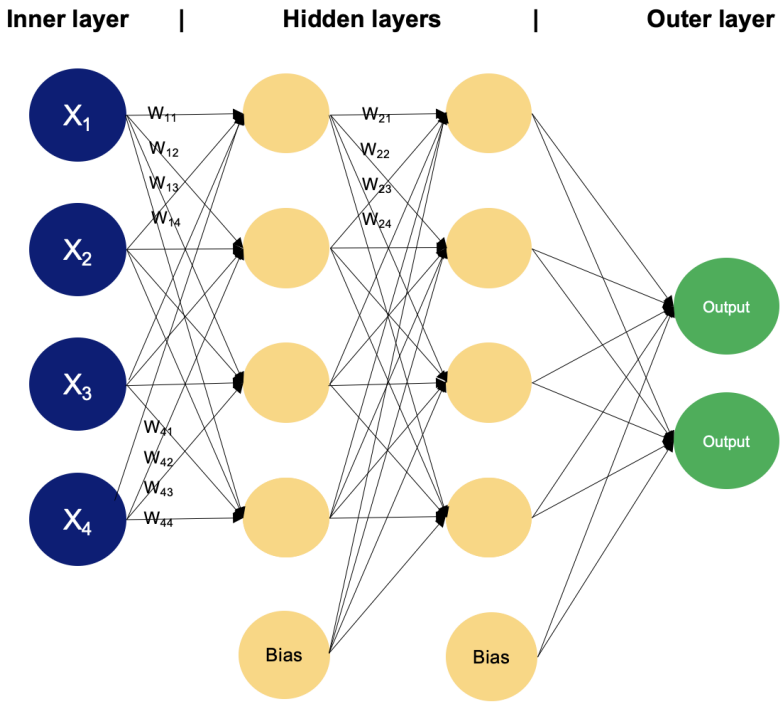
- 5 . Randoms Forests

Model and Result

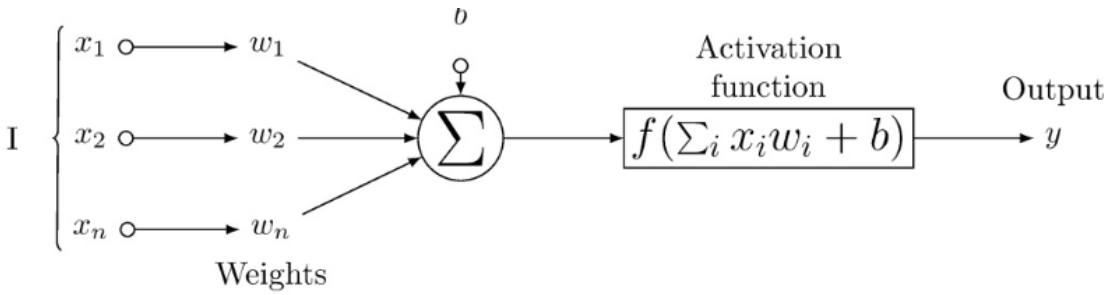


III. Methodology

- 6.Neural Networks MLP (Multi-layer Perception)
- A type of Artificial Neural Network with fully connected dense layers, transform any input dimension to the desired dimension.



- Input layer: nodes of neurons that receive the data
- Hidden layer: Each neuron in a hidden layer receive Inputs from the previuos layer and produces an output
- Output layer: Neurons that produce the final output of the network



III. Methodology

- Hyperparameters:
  +) Learning rate: determines how quickly the code update the weight values.
- +) Number of hidden layers and nodes:
- +) Activation function: Used to weight sums of inputs
- +) Loss function: measure how well ANN model fits the dataset
- +) Optimizers: update the weight parameters to minimize the loss function.

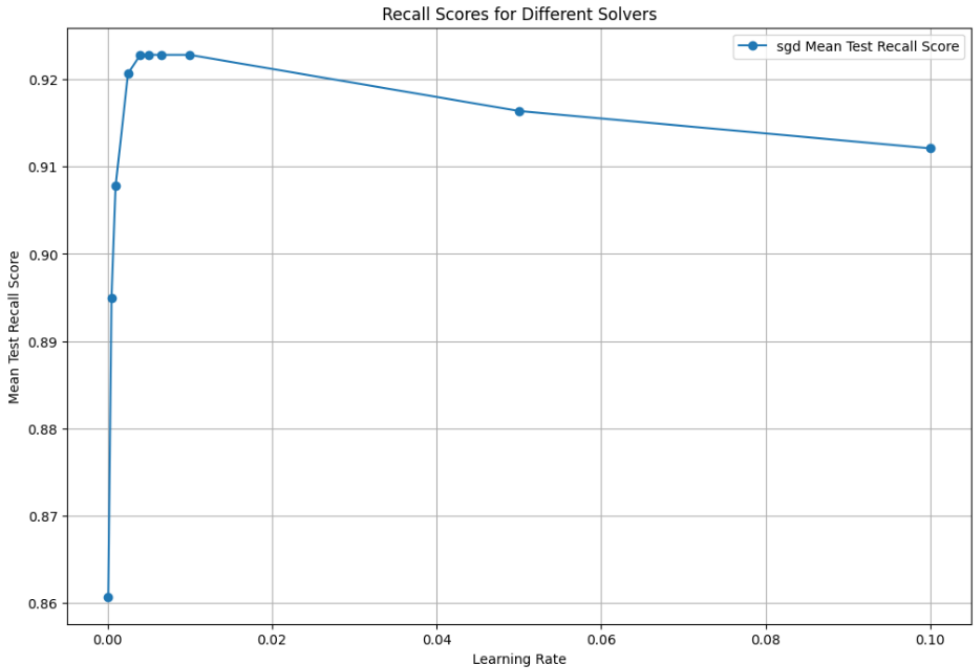
$$ReLU$$

$$f(x) = max\,(0, x)$$

$$L_{BCE} = -\frac{1}{n}\sum_{i=1}^{n}(Y_i \cdot \log \hat{Y}_i + (1 - Y_i) \cdot \log (1 - \hat{Y}_i))$$

III. Methodology

- Hyperparameters tuning:



Graph: learning rate of models with accuracy metric

Number of hidden layers		(40,20,10,5)	(100,50)
Training Set	Accuracy	0.99988	0.99966
	Precision	0.99976	0.99933
	F1 Score	0.99988	0.99966
	Recall	1.00000	1.0000
Test Set	Accuracy	0.99906	0.99306
	Precision	0.64285	0.64285
	F1 Score	0.69230	0.69230
	Recall	0.75000	0.7500

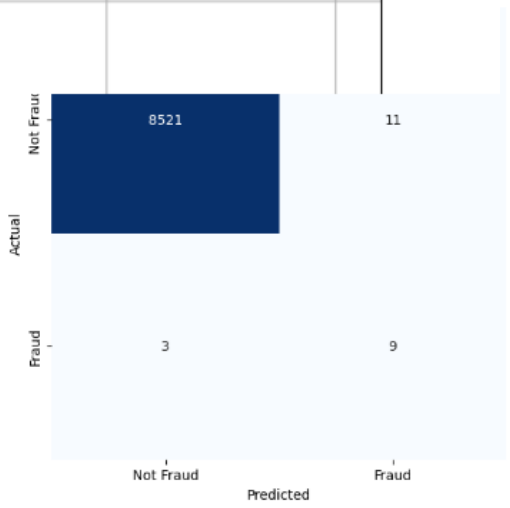
III. Methodology

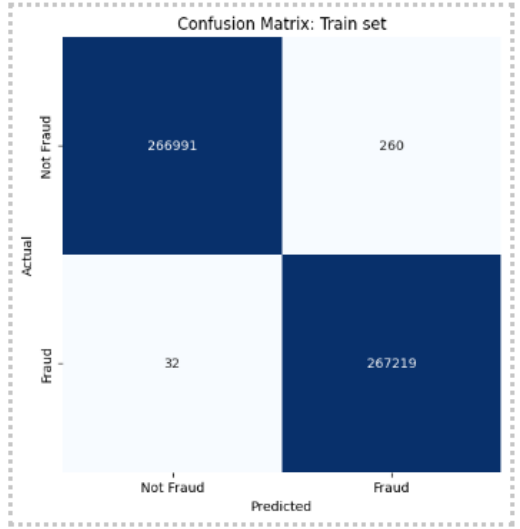
```
Best parameters found:
 {'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes': (40, 20, 10, 5),
'learning_rate': 'adaptive', 'learning_rate_init': 0.001, 'solver': 'sgd'}
```

	Validation Results	Testing Results
Accuracy	0.9987	0.9989
Precision	0.5555	0.64
Recall	0.7692	0.75
F1 score	0.6451	0.6669
ROC AUC	0.9116	0.9218

III. Methodology

- 



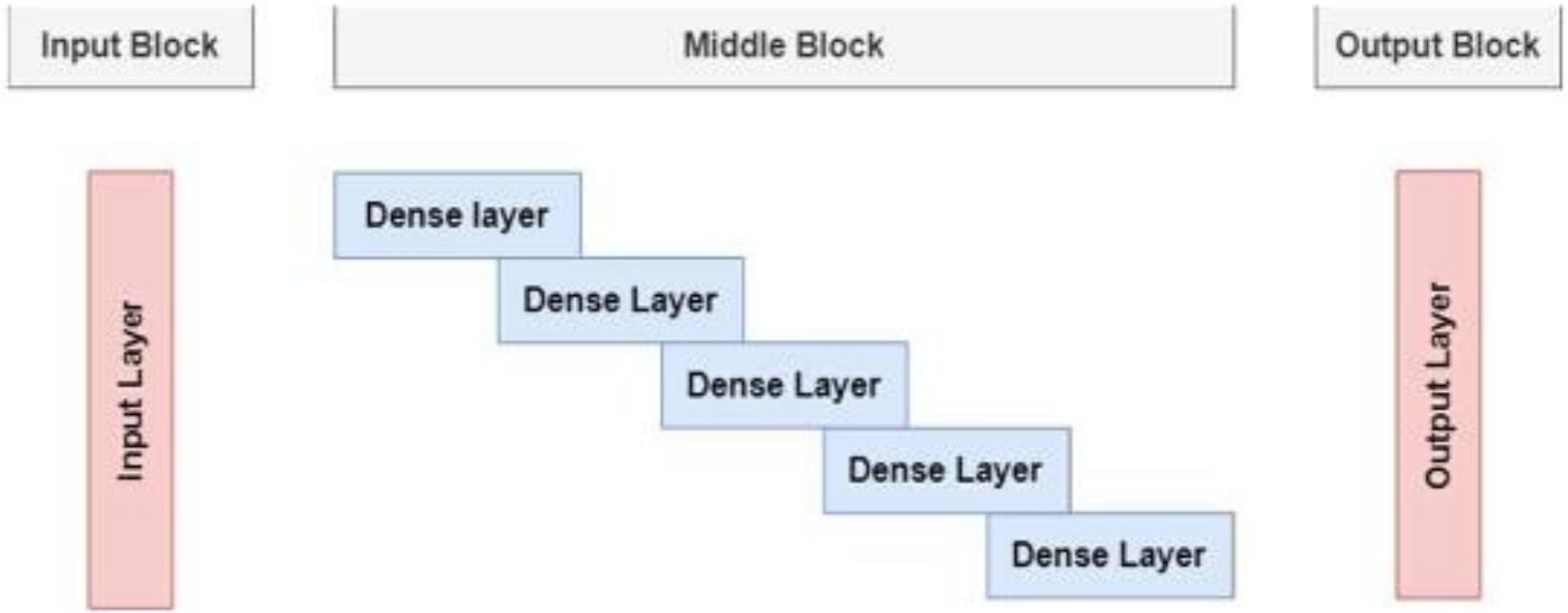


Picture: the confusion matrix for Testing data and training data

III. Methodology

- Deep Neural Network (DNN)
- Deep Neural Networks (DNNs) are a subset of artificial neural networks (ANNs) with multiple layers between the input and output layers, also known as hidden layers, allow the network to model complex relationships in data.
- **Key Components of DNNs**:
- +) Input Layer: Receives the initial data for processing.
- +) Hidden Layers: Intermediate layers that transform the input into a more abstract representation.
- +) Output Layer: Produces the final prediction or classification.
- +) Activation Functions: Functions like ReLU (Rectified Linear Unit) and Sigmoid that introduce non-linearity, enabling the network to learn complex patterns.
- +) Optimization Algorithm: Techniques like Stochastic Gradient Descent (SGD) that adjust the weights to minimize the loss function.

III. Methodology



A Deep Neural Network Structure

III. Methodology

- Deep Neural Network (DNN)

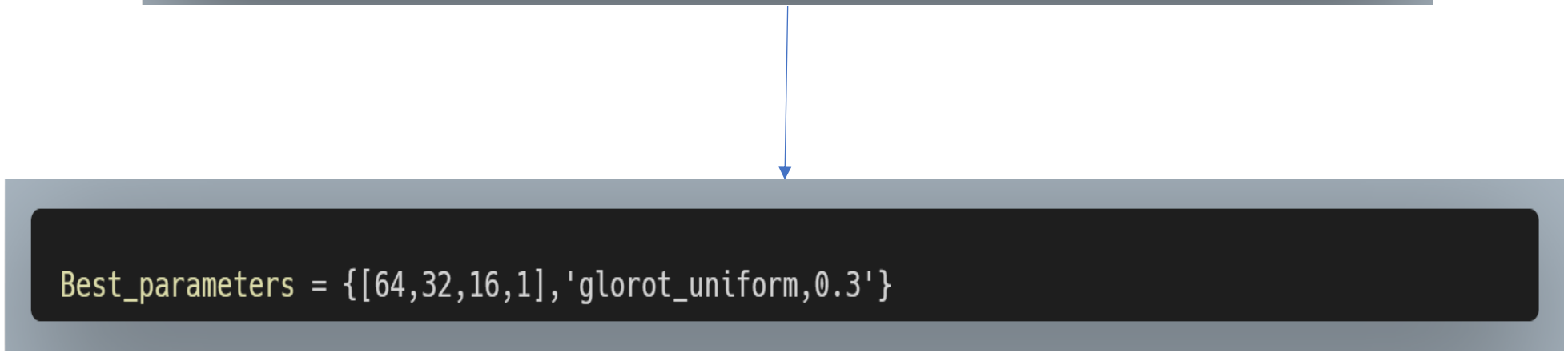
```
def create_model():
    model = Sequential()
    model.add(Dense(64, input_dim=X_train.shape[1], activation='relu',
kernel_initializer=GlorotUniform()))
    model.add(Dense(32, activation='relu', kernel_initializer=GlorotUniform()))
    model.add(Dense(16, activation='relu', kernel_initializer=GlorotUniform()))
    model.add(Dense(1, activation='sigmoid', kernel_initializer=GlorotUniform()))

    optimizer = SGD(learning_rate=0.01)
    model.compile(optimizer=optimizer, loss='binary_crossentropy', metrics=['accuracy'])
    return model
```

III. Methodology

- Deep Neural Network (DNN)
- Hyperparameter tuning:

```
# Initial weights to test
initial_weights_options = ['glorot_uniform', 'he_normal', 'lecun_normal']
# Network sizes to test
network_sizes = [[64],[32],[64,32],[32,16],[64, 32, 16],[64,32,16,8],[128,64,32,16,8],
[64,32,16,8,4]]
# Learning rates to test
learning_rates = [0.001,0.005,0.01,0.03,0.05,0.08,0.1,0.2,0.3,0.5]
```

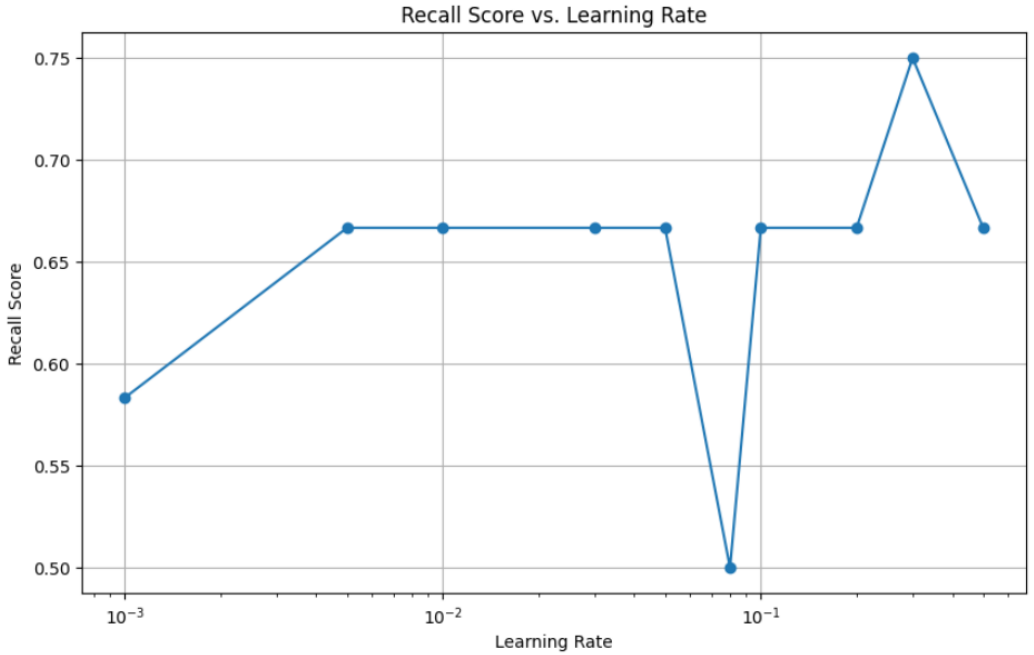


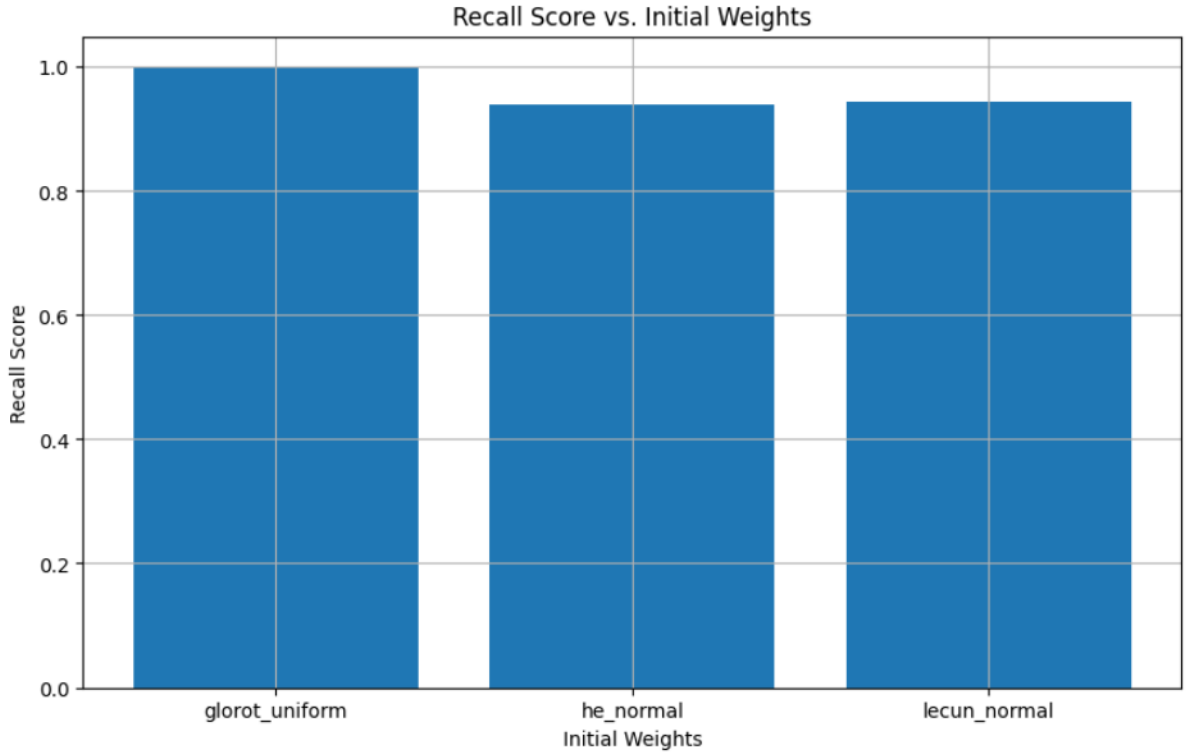
```
Best_parameters = {[64,32,16,1],'glorot_uniform,0.3'}
```

III. Methodology

Hyperparameter tuning

Learning rate	0.001	0.005	0.01	0.03	0.05	0.08	0.1	0.2	0.3	0.5
Precision	0.88	0.89	0.89	0.89	0.89	0.86	0.89	0.89	0.9	0.89

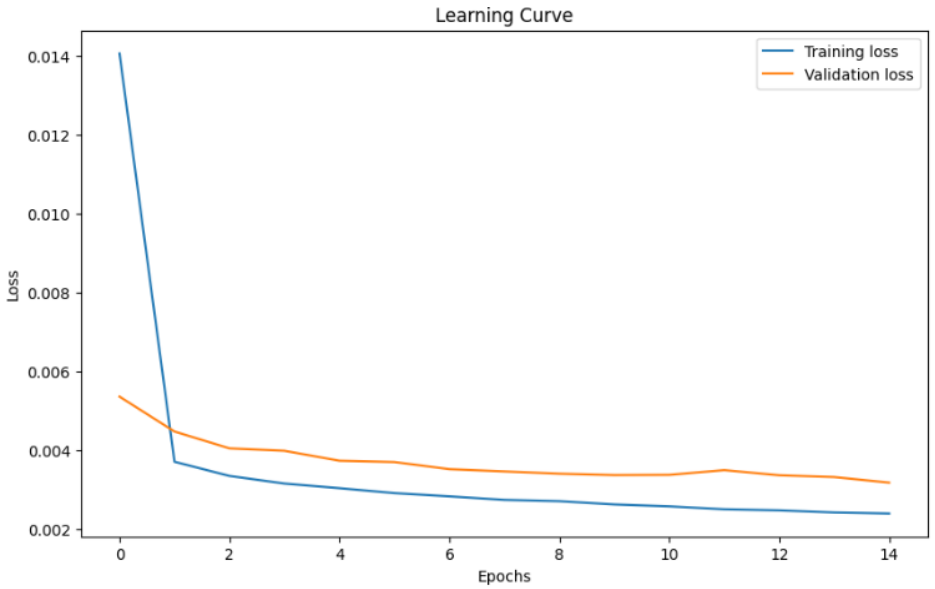




III. Methodology

Results:

Metrics	Testing Results
Accuracy	0.98
Precision	0.96
Recall	0.89
F1 Score	0.90
ROC AUC	0.950



Graph: the learning curve of DNN

HUST

Table of contents

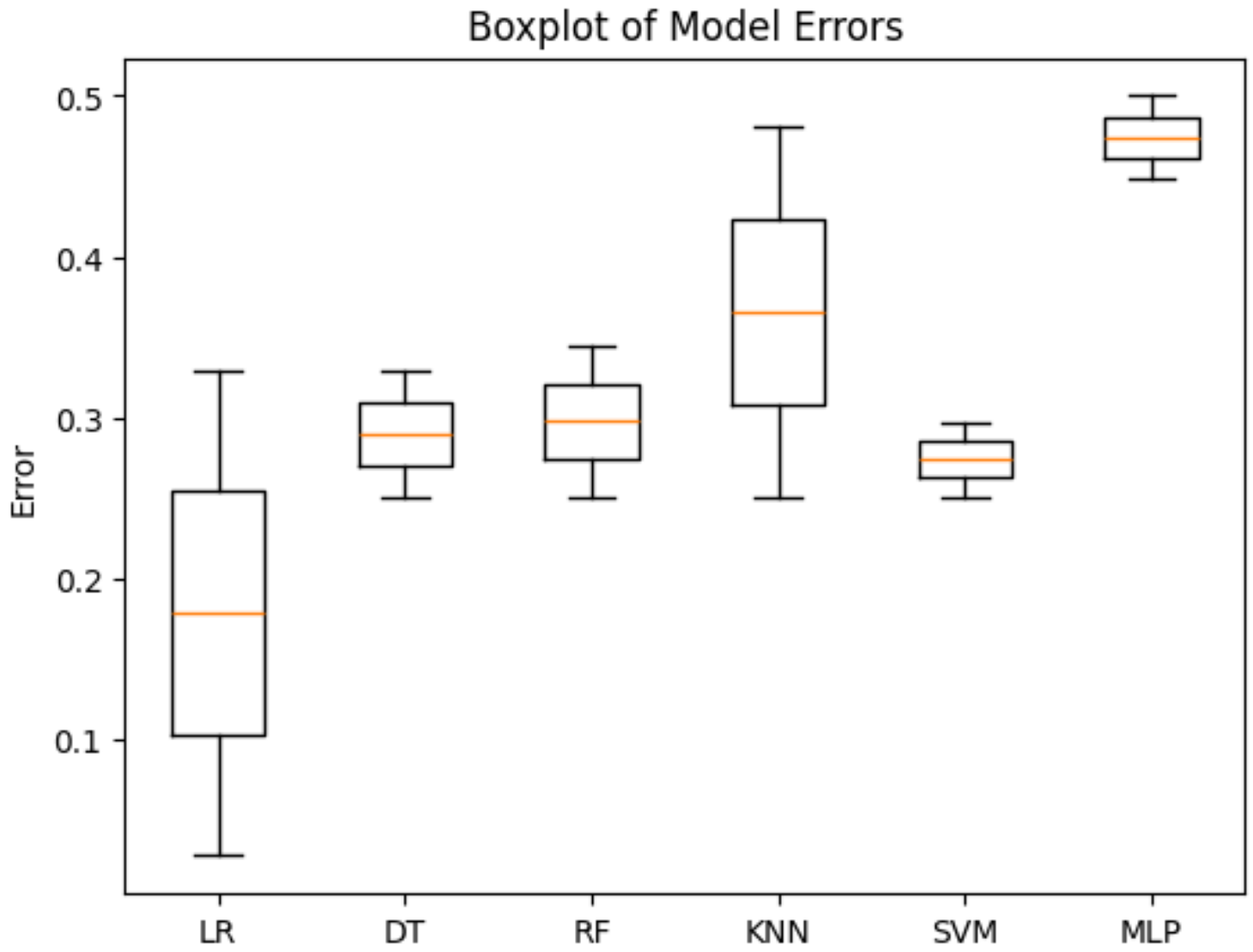
IV.Results and comparisons

- Results:

Methods	Accuracy	Precision	F1-Score	Recall
LR	0.96	0.93	0.96	0.98
SVM	0.95	0.95	0.94	0.95
DTree	1.0	1.0	1.0	1.0
RF	1.0	1.0	1.0	1.0
KNN	0.95	0.98	0.91	0.84
ANN	0.9989	0.64	0.6669	0.75
DNN	0.90	0.90	0.82	0.75

IV.Results and comparisons

- Results:



IV.Results and comparisons

- Conclusion
- Overall, to compare our seven ways of approaching this credit card fraud classification method, we use comparisons, and it is proven that the Decision Tree, Random Forest perform better than the other models while Artificial Neural Network perform worse than other models.
- This can be explained, the Artificial Neural Network is likely to be overfitted while models such as Decision Tree and Random Forest are good at handling imbalance data and classification problem.

IV.Results and comparisons

- Future Implementation
- The project currently relies on a data set with comprehensive information about customers and their transactions. To address this, the model can be adapted to work with more limited and anonymized data, making it more applicable to real-world conditions.
- Some techniques can still be applied to enhance the model's performance. Techniques such as the BayesSearch for hyperparameter tuning process, Adaptive Synthetic Sampling to handle the imbalanced dataset,.
- The project is still not user-friendly. A better user interface system can be built.
- In conclusion, the capstone project was able to perform and give out satisfactory results but there is still many improvements that can be made to further improve the performance of the project.

HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY