Report: Summary of the results of classification techniques applied to banking transaction data

AFERHANE Jamila

March 20, 2023

Contents

Introduction:

The purpose of this report is to present the results of applying four different algorithms - logistic regression, KNN, SVM, and DNN - to a banking transactions dataset for fraud detection. Each algorithm is evaluated based on a variety of performance metrics, including precision, recall, F1 score, and AUC. The goal of this analysis is to determine which algorithm is best suited for detecting fraud instances in banking transactions.

Machine Learning:

In this section, we describe the application of machine learning techniques to our Banking transaction dataset to develop a robust model for predicting whether a transaction is fraudulent or not.

Specifically, we implemented three models- Logistic Regression, Support Vector Machines (SVM), and K-Nearest-Neighbors (KNN)- and evaluated their performance to determine the best-fit model.

In our code, we begin by importing the required libraries and the dataset. We then organize and split the data into training and testing sets. For each technique, we train the model on the training data and make predictions on the testing data. Concerning evaluation, we use metrics such as accuracy score, confusion matrix, and receiver operating characteristic (ROC) curve.

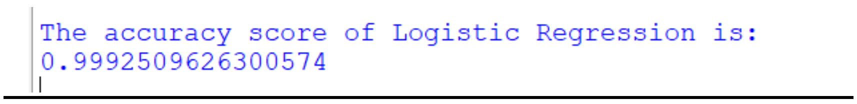
Remarque:

- In all our models, we set the random state to a fixed value (0). So, compiling the code will give similar results.
- All the pictures in this report (except the explanatory graph of ROC curve) are taken from the compilation of the provided code "***transactions_classification.py***"

Logistic Regression:

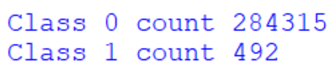
We apply our trained Logistic Regression model on the testing set (**30%** of our data) and we get the following results:

Accuracy score: It presents the percentage of correctly predicted labels (=Number of correctly predicted labels/Total number of labels):



```
The accuracy score of Logistic Regression is:
0.9992509626300574
```

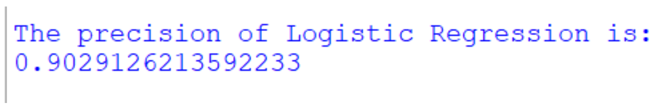
99.925 % as the accuracy score is considered to be **very high** which means that the model is performing very well on the given data. But, having **imbalanced data** as we have (28 4315 for "0" and 492 for "1") made the accuracy **score not a reliable metric** for evaluation.



```
Class 0 count 284315
Class 1 count 492
```

For this we use:

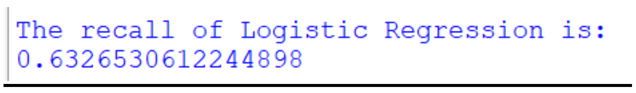
Precision: It measures the ability of the model to correctly identify positive instances (= **true positives/ (true positives + false positives)**):



```
The precision of Logistic Regression is:
0.9029126213592233
```

As it is shown in the picture above, the precision score is considered **high and satisfactory**; the model can correctly identify **90.29%** of the positive instances and **avoid falsely predicting negative** instances as positive.

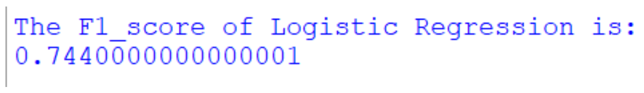
Recall (or positive rate): It measured the ability of the model to correctly identify all positive instances (= **true positives / (true positives + false negatives)**)



```
The recall of Logistic Regression is:
0.6326530612244898
```

A recall of 0.6327 indicated that the model is **missing a significant number of positive instances**. **36.73%** are classified as non-fraudulent transactions whereas they are fraudulent, which may be catastrophic for a bank.

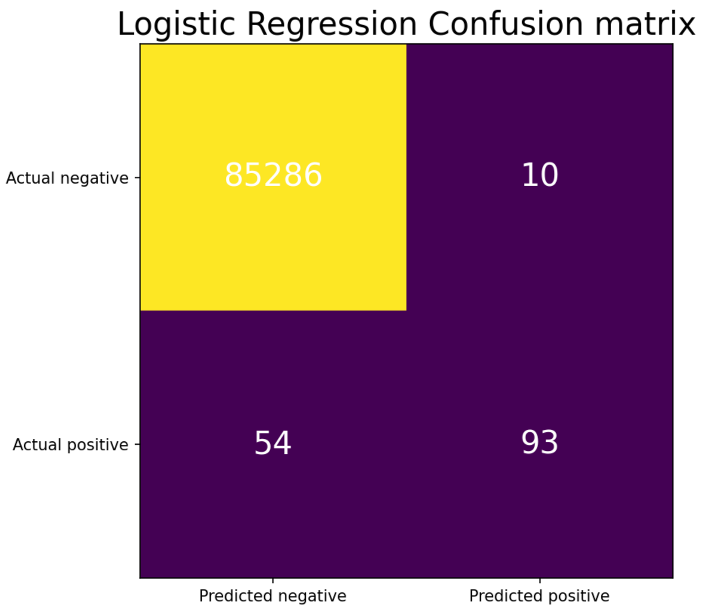
F1 score: It is a useful metric for evaluating the performance of models on imbalanced datasets. It balances both precision and recall and provides a single score that summarizes the overall performance of the model (= **2(precision*recall) / (precision + recall)**)



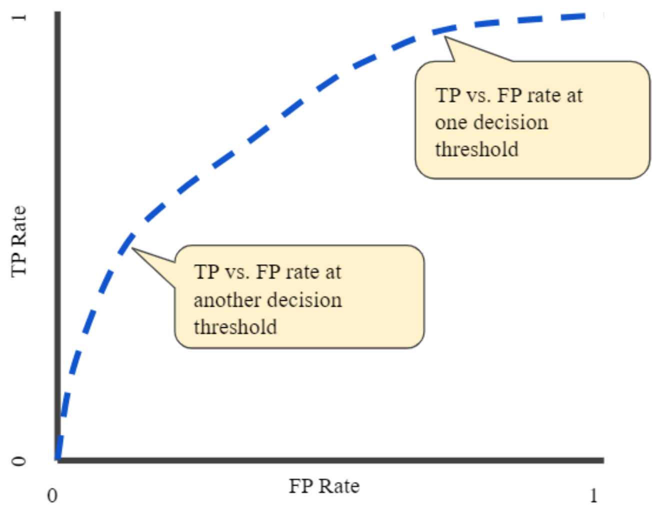
```
The F1_score of Logistic Regression is:
0.7440000000000001
```

For fraud detection, an F1 score of 0.744 is a **reasonably good score**.

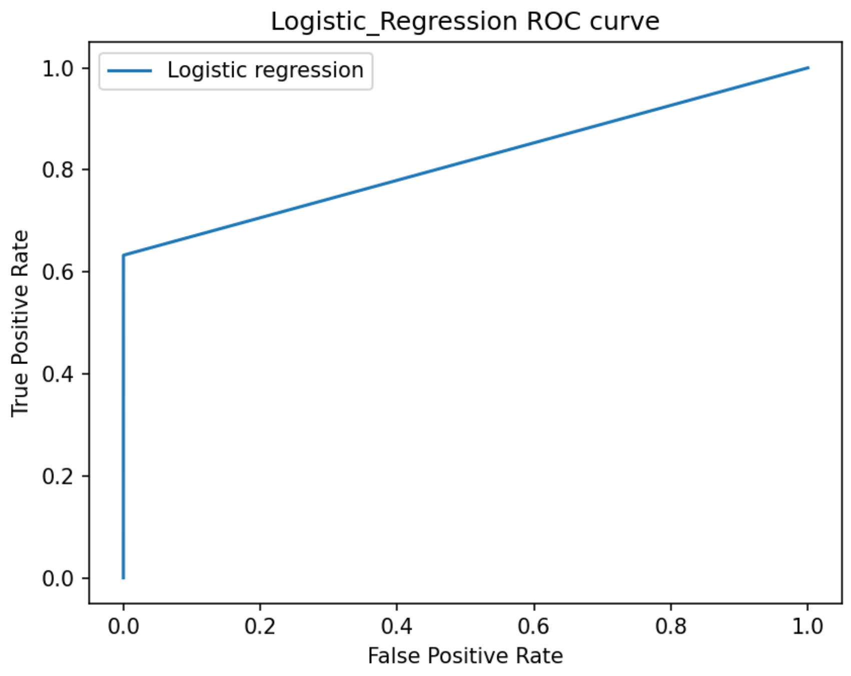
Confusion matrix: It is a table that shows the true positive, true negative, false positive, and false negative values for a binary classification model. For our model and dataset, we get the following confusion matrix:



ROC curve: It is a graphical representation of the performance of a binary classifier, which measures the trade-off between the **True Positive Rate** (TPR= true positive / (true positive + false negative) and **False Positive Rate** (FPR= false positive / (false positive + true negative)) for different **classification thresholds** (can be thought of as the decision boundary of the sigmoid function in logistic regression). Here is an explanatory graph:


TP Rate
1
0
0
FP Rate
1
TP vs. FP rate at one decision threshold
TP vs. FP rate at another decision threshold

For our model, we get the following ROC curve:



The closer the curve is to the top-left corner of the plot, the better the classifier is at correctly identifying positive cases while minimizing false positives.

To quantify the performance of the classifier using the ROC curve, we identify the AUC:

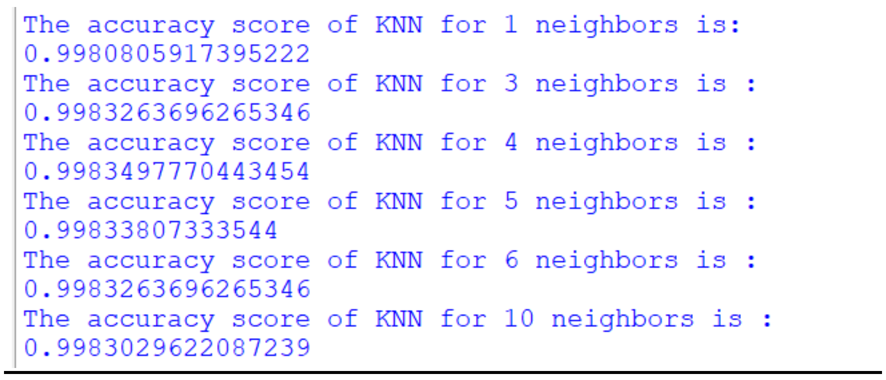
Area Under the Curve AUC: It refers to the area under the ROC curve. A higher AUC corresponds to a better classifier performance:

```
The AUC of Logistic Regression is:
0.8162679112162592
```

K-Nearest Neighbors (KNN):

After training our KNN model, we apply it to our testing set and analyze the outcomes to evaluate its performance. The following results have been obtained:

Accuracy score: We trained the model for different k (k=1,2,3,4,5,6,10) and we measured the accuracy score for each one. We obtain the following results:

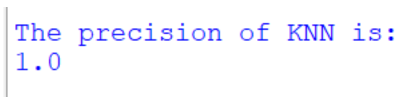


```
The accuracy score of KNN for 1 neighbors is:
0.9980805917395222
The accuracy score of KNN for 3 neighbors is :
0.9983263696265346
The accuracy score of KNN for 4 neighbors is :
0.9983497770443454
The accuracy score of KNN for 5 neighbors is :
0.99833807333544
The accuracy score of KNN for 6 neighbors is :
0.9983263696265346
The accuracy score of KNN for 10 neighbors is :
0.9983029622087239
```

It seems like **k=4 is the best choice** for our dataset since it has the highest score.

The rest of the metrics are measured for **k=4.**

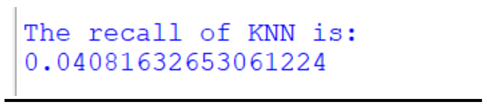
Precision: = true positives/ (true positives + false positives)



```
The precision of KNN is:
1.0
```

The precision score is **1** which means that the model **did not make any false positive predictions** and all the predicted positive instances were positive. But a precision score of 1 doesn't necessarily mean that the model is perfect, as the other metrics show.

Recall (or positive rate): = true positives / (true positives + false negatives)

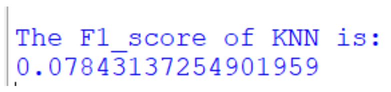


```
The recall of KNN is:
0.04081632653061224
```

The model was able to correctly identify only **4%** of the positive instances in the ground truth data. It means that the **model is missing a large number of positive instances**.

In our case of fraud detection, we can say that **KNN is not suitable for our dataset.**

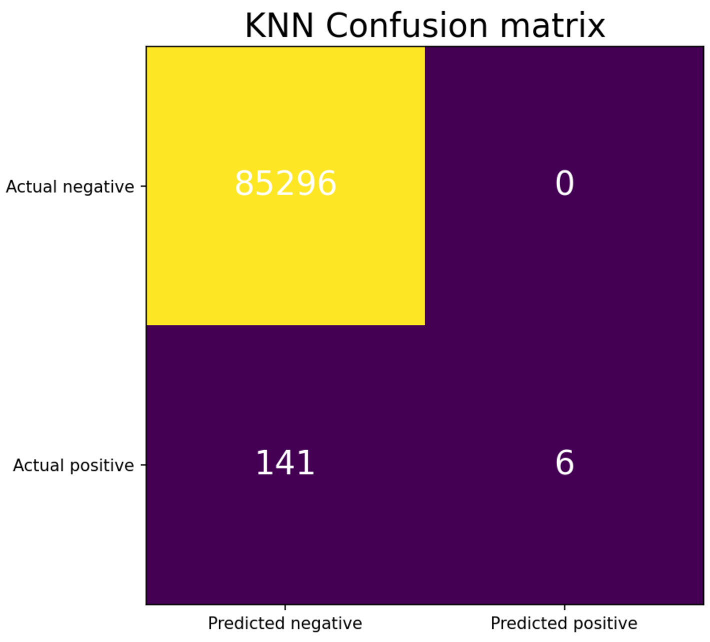
F1 score: = 2(precision*recall) / (precision + recall)



```
The F1_score of KNN is:
0.07843137254901959
```

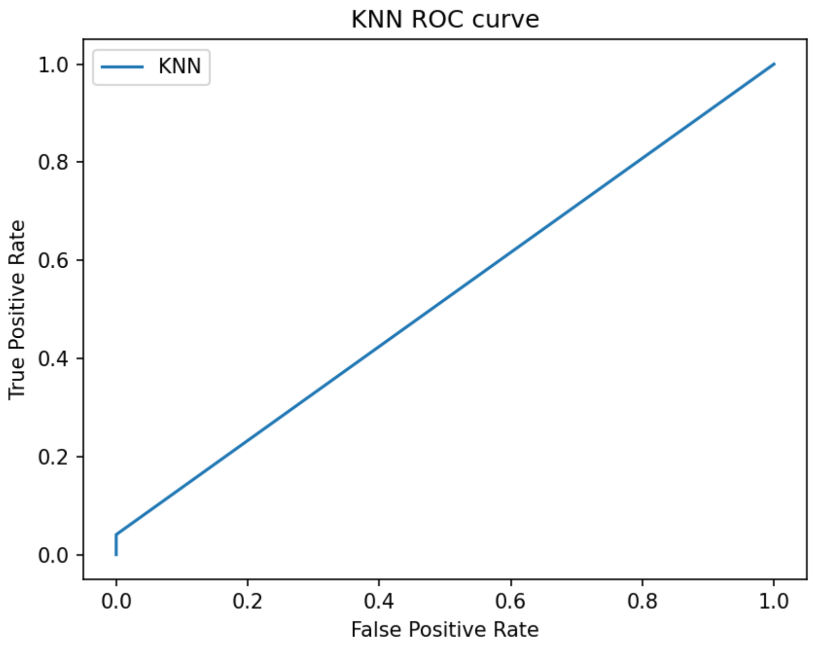
The obtained score indicates that the model is **performing poorly in the positive class**, which could potentially be a concern for a bank that needs to accurately detect fraudulent activities in its operations.

Confusion matrix:



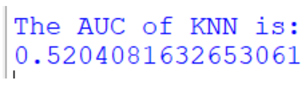
A total of **141** instances out of **147** are incorrectly predicted as negative by the model

ROC curve:



For our case of fraud detection, a ROC curve close to the line y=x is not ideal since the cost of false negatives (missed fraudulent transactions) can be very high.

Area Under the Curve AUC:



```
The AUC of KNN is:
0.5204081632653061
```

This score suggests that the model's ability to separate the positive and negative classes is only slightly better than guessing.

Conclusion: The KNN model is not performing well on the task of fraud detection

Support Vector Machines (SVM):

We found that training the SVM model using a linear kernel and a capacity of 100 gave a good performance. Due to this reason and the extended training time required, we did not explore other parameters. Here are the founded metrics:

Accuracy score:

```
The accuracy score of SVM is:
0.9987477031471274
```

This model gives a high accuracy score of **99.87%**, but it is still less than the Logistic Regression score (99.92%)

Precision: = true positives/ (true positives + false positives)

```
The precision of SVM is:
0.7702702702702703
```

According to this precision score, the model is correctly identifying 77.02% of the positives. In other words, when the model classifies an instance as positive, there is a 77% chance that the classification is correct.

Recall (or positive rate): = true positives / (true positives + false negatives)

```
The recall of SVM is:
0.3877551020408163
```

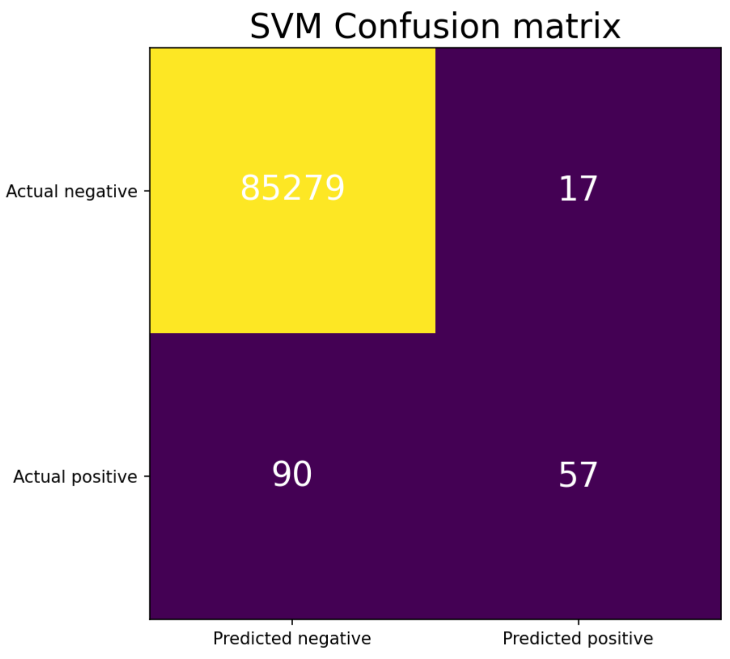
The model is missing approximately **62%** of positive instances.

F1 score: = 2(precision*recall) / (precision + recall)

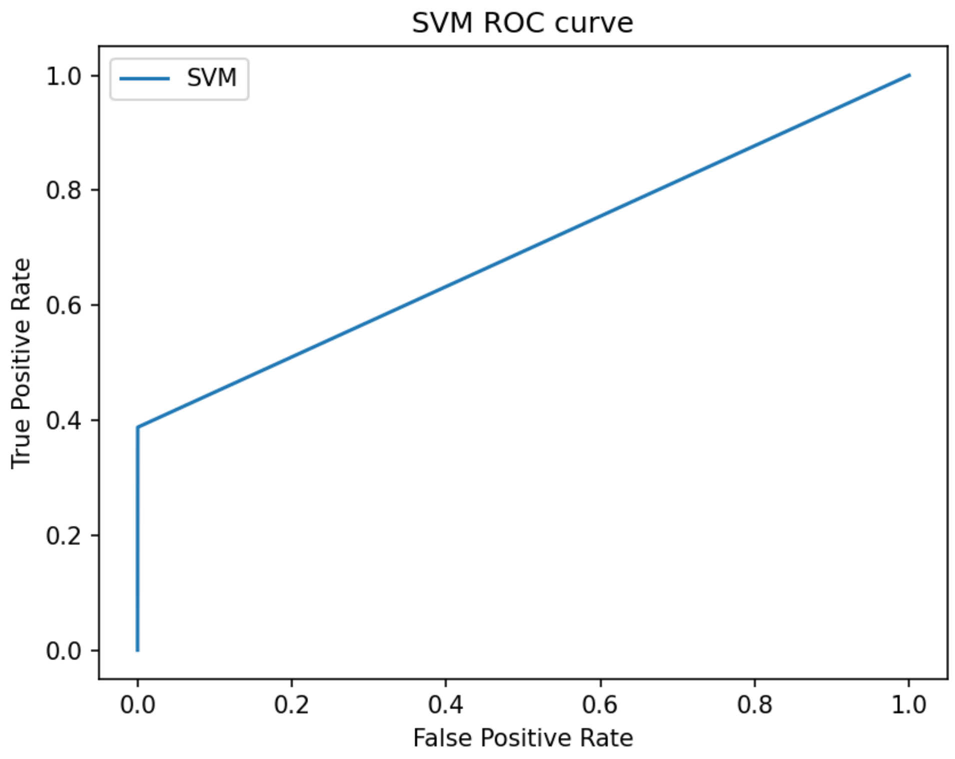
```
The F1_score of SVM is:
0.5158371040723981
```

An F1 score of **0.5153** means that the model has achieved a **balance** between **precision** and **recall**.

Confusion matrix:

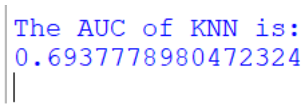


ROC curve:



This curve is better than the **KNN ROC curve** but is still not perfect.

Area Under the Curve AUC:



```
The AUC of KNN is:
0.6937778980472324
```

An AUC score of **0.69** indicates that the performance of the model is slightly better than random guessing (an AUC score of 0.5).

Conclusion:

After analyzing the performance of three different models on our banking transactions dataset, we determine that logistic regression is the most suitable model for our purposes. This conclusion is based on the evaluation of various metrics, including precision, recall, and F1 score.

Logistic regression outperformed the other models in **detecting positive instances**, with a **Recall** score of **0.632**. In comparison, the Recall scores for SVM and KNN were 0.38 and 0.04, respectively.

Deep Learning:

In the following section, we will employ a deep learning technique known as **Deep Neural Network classifier**.

DNN:

To implement our deep neural network classifier, we construct a neural network model consisting of three layers with 1000 epochs and 5000 steps. The model is trained on our banking transactions dataset. During the training process, the model learns to recognize patterns and relationships within the data that can be used to identify potential instances of fraud.

To evaluate the performance, we test the model's ability to accurately classify new data by applying it to the testing set.

Here are the results of the model predictions:

<u>Accuracy score:</u>

```
The accuracy score of DNN is:
0.9981274065751436
```

A score of **0.998** for fraud detection is a **high accuracy** and suggests that the DNN is able to accurately classify the vast majority of the data points.

<u>Precision</u>: = true positives/ (true positives + false positives)

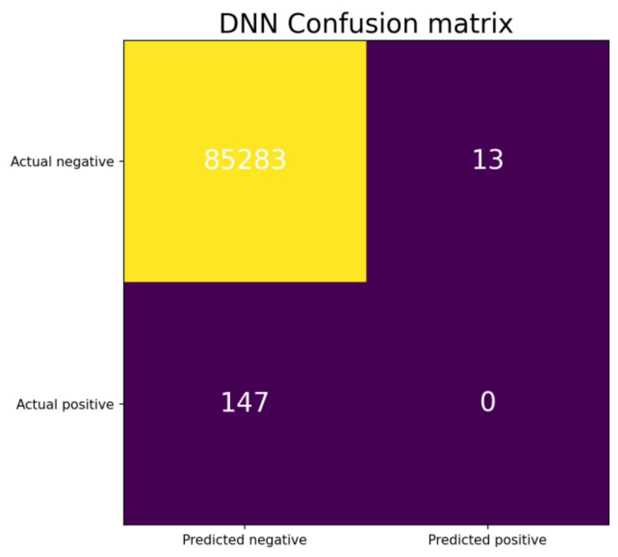
```
The precision of DNN is:
0.0
```

A precision of 0 means that the model did not correctly identify any of the fraudulent transactions. We can say that DNN is a bad method for fraud detection for our dataset.

<u>Recall (or positive rate):</u> = true positives / (true positives + false negatives)

The confusion matrix shows that more clearly:

<u>Confusion matrix:</u>

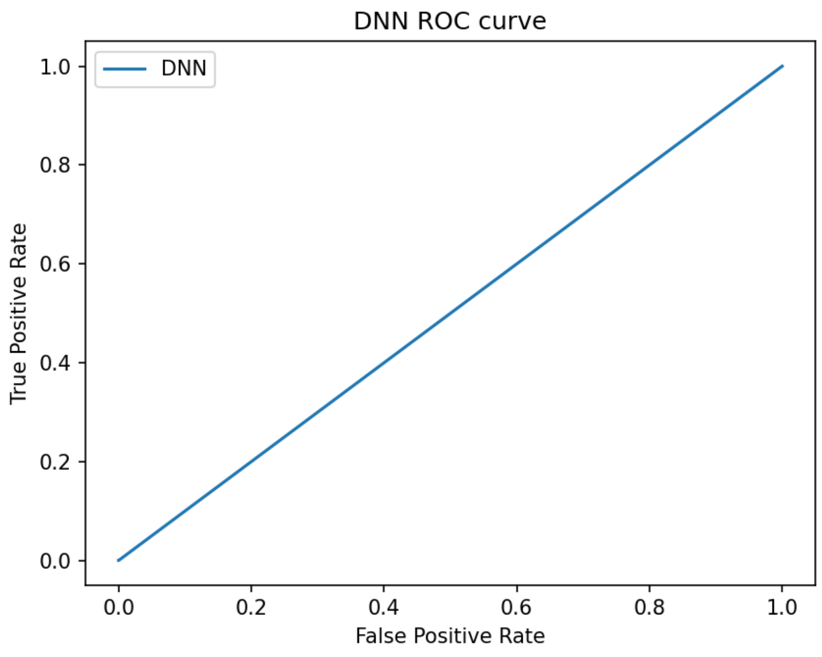


None of the actual positives are predicted as positive. DNN is a useless technique in this case.

So, it is obvious that **the Recall** and **the F1_score** are also equal to 0:

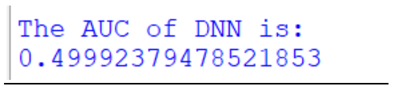
```
The recall of DNN is:
0.0

The F1_score of DNN is:
0.0
```

ROC curve:



Having a ROC curve of y=x as an equation means that the model has no ability to differentiate between fraudulent and non-fraudulent transactions.

AUC score:



```
The AUC of DNN is:
0.49992379478521853
```

This score confirms that the DNN model is no better that the random chance

Conclusion:

To conclude, Logistic Regression is the best model for fraud detection with our dataset among the models tested, while the DNN performed the worst with no correct positive predictions. The logistic regression model predicted 93 positive instances out of 147, while SVM predicted 57 and KNN only found 6 positive instances.

It is worth saying that the winner is the Machine Learning technique, which shows that Deep learning is not always the best choice for building robust models.