ML Model for Credit Card Fraud Detection

❖ Table of content

- Introduction
- Data Description
- Exploratory Data Analysis (EDA)
- Feature Engineering
- Model Selection and Training
- Conclusion

➢ Introduction: -

Problem Statement:

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

➢ Data Description: -

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

➢ Exploratory Data Analysis (EDA) :-

Exploratory Data Analysis (EDA) plays a crucial role in understanding the dataset and preparing it for machine learning tasks such as credit card fraud detection. Effective EDA not only helps in understanding the characteristics and quality of the dataset but also informs preprocessing steps and feature engineering strategies essential for developing accurate fraud detection models. It forms the foundation for making informed decisions throughout the machine learning pipeline, ensuring that the model is trained on reliable and relevant data.

➢ Feature Engineering: -

Effective feature engineering in credit card fraud analysis involves a combination of domain expertise, data exploration, and creative thinking to extract meaningful information from raw transaction data. It aims to enhance the discriminatory power of machine learning models and improve their ability to detect fraudulent activities accurately while minimizing false positives.

. Model Selection: -

Machine Learning Types: -

Supervised Machine Learning Classifiers In this step, we will describe some supervised machine learning classifiers named Logistic Regression, k-nearest neighbors, Support Vector Machine.

1. Logistic Regression (LR) :-

Logistic Regression (LR) is a supervised machine learning data classification algorithm that mines real-valued features from the input, multiplies each of them by a weight, adds them, and transfers the sum through a sigmoid function to produce a probability. A threshold is used to finalize a decision [2]. A solution for classification of our data set is LR which Instead of fitting a straight line or hyperplane uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$\text{Logistic}(\eta) = 1/(1 + \exp(-\eta))$$

As η goes from −∞ to ∞, logistic (η) goes from 0 to 1, a “squashing function”. In our study, we used a maximum 4000 iterations to converge the output.

The Evaluation we received after using this model is as shown below.

```
Model Evaluation
```

```
Accuracy score
```

[80]:
```
# accuracy on training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

[82]:
```
print('Accuracy on Training data : ', training_data_accuracy)
```

```
Accuracy on Training data :  0.9415501905972046
```

[84]:
```
# accuracy on test data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

[86]:
```
print('Accuracy score on Test Data : ', test_data_accuracy)
```

```
Accuracy score on Test Data :  0.9187817258883249
```

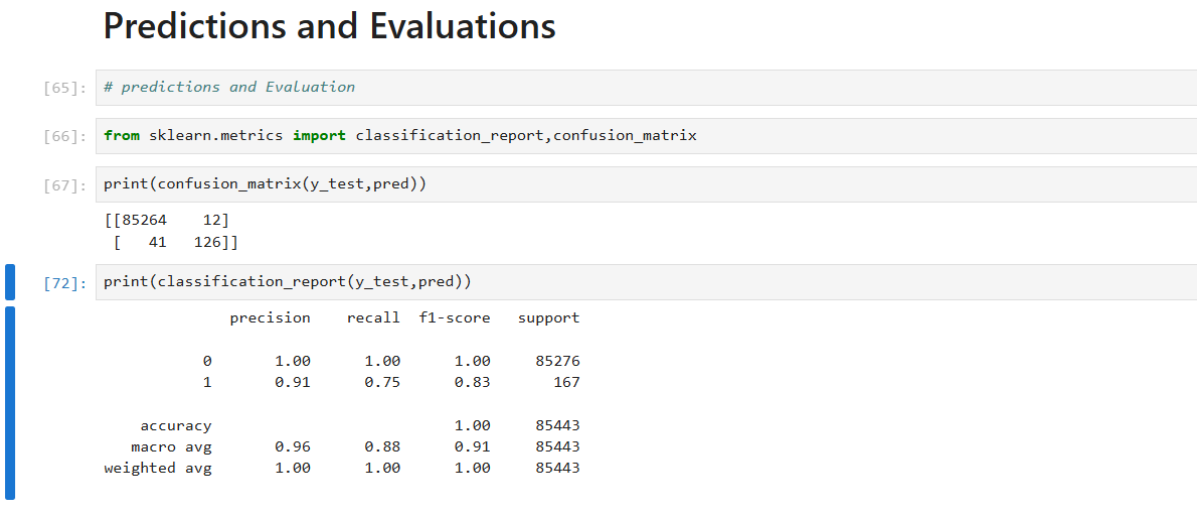
2. k-nearest neighbors (KNN) **:-**

(KNN) is a non-parametric process we used for diabetic data classification. In KNN a data is classified by a majority vote of its

neighbors, with the data being allotted to the class most mutual amongst its K nearest neighbors estimated by a distance function. If K = 1, then the data is simply allotted to the class of its nearest neighbor. KNN algorithm is as below :

Algorithm 1 KNN :-
1: Let m be the number of training data samples. Let p be an unknown point that needs to be classified
2: Storing the training samples in an array of data points arr[]. Each element of this array denotes a tuple (x, y).
3: for i = 0 to m do
4: Calculating distance d(arr[i], p)
5: end for
6: Making set S of K smallest distances achieved. Each of these distances resembles an already classified data point
7: Returning the majority label among S

The Evaluation we received after using this model is as shown below.



Predictions and Evaluations

```
[65]: # predictions and Evaluation

[66]: from sklearn.metrics import classification_report,confusion_matrix

[67]: print(confusion_matrix(y_test,pred))

[[85264    12]
 [   41   126]]

[72]: print(classification_report(y_test,pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85276
           1       0.91      0.75      0.83       167

    accuracy                           1.00     85443
   macro avg       0.96      0.88      0.91     85443
weighted avg       1.00      1.00      1.00     85443
```

3. Support Vector Machine (SVM) :-

A Support Vector Machine is a discriminative classifier well-defined by a separating hyperplane. In other words, specified labelled training data,

the algorithm generates an optimal hyperplane which classifies the new data point. In two dimensional space, this hyperplane is a line separating a plane in two parts where each class lay in either side. Along with linear data it also classifies the non-linear data using kernel trick. Hyperplane can be written as the set of points $\vec{x}$ satisfying:

$$\vec{w}.\vec{x} - b = 0$$

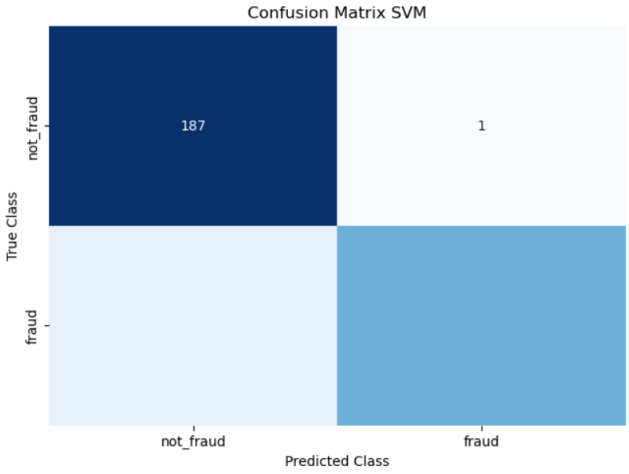
The parameter $\frac{b}{\|\vec{w}\|}$ defines the offset of the hyperplane from the origin along the vector $\vec{w}$ which needs to be maximize. SVM uses ”regularization parameter” which controls the trade-off between experimental error and complexity of the assumption space used

The Evaluation we received after using this model is as shown below.

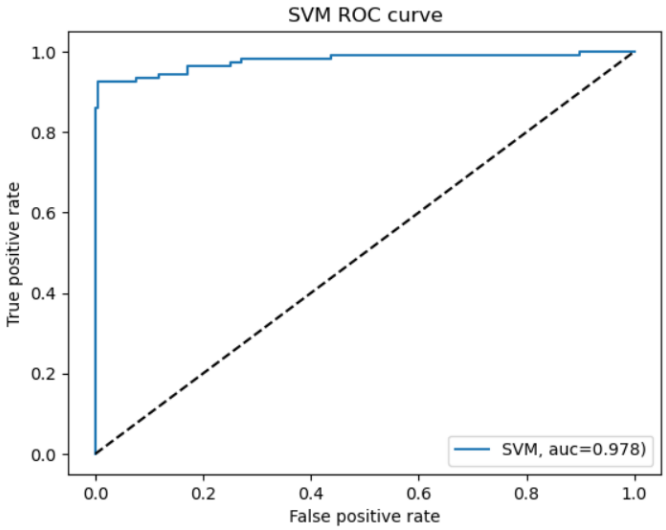
```
[52]: #scores
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
```

```
Accuracy SVM: 0.9459459459459459
Precision SVM: 0.9893617021276596
Recall SVM: 0.8611111111111112
F1 Score SVM: 0.9207920792079208
```

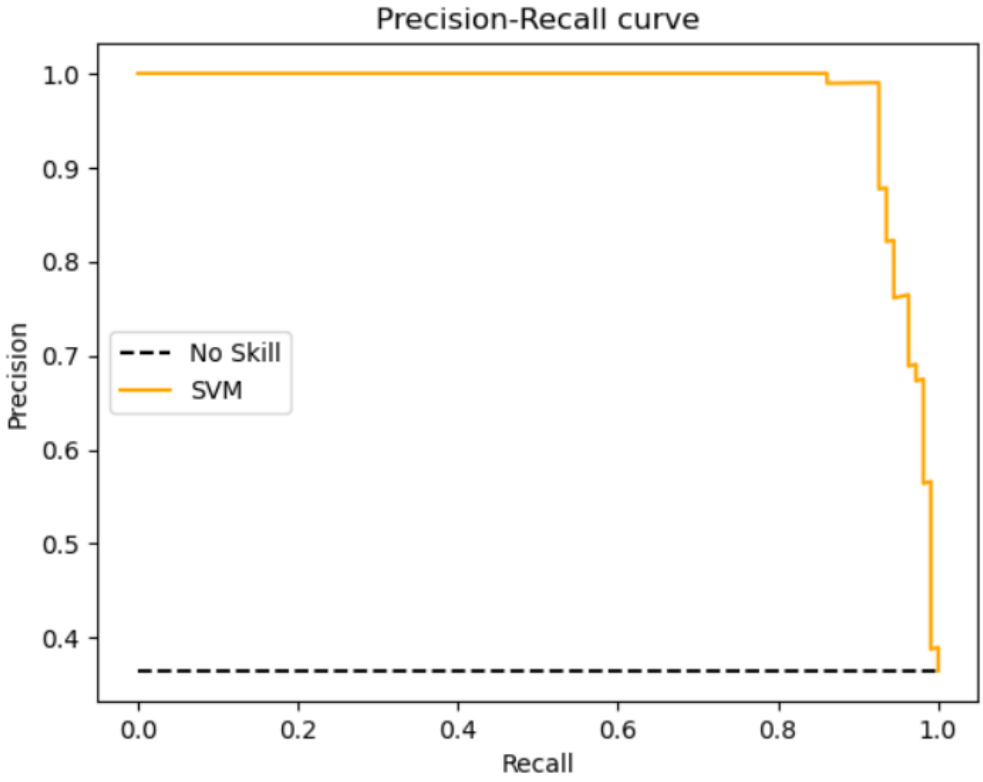
Confusion Matrix



ROC Curve :-



Precision Recall curve: -



Conclusion :-

1. SVM Model:

- **Accuracy (SVM):** 0.9459
- **Precision (SVM):** 0.9894
- **Recall (SVM):** 0.8611
- **F1 Score (SVM):** 0.9208

2. Logistic Regression Model:

- Accuracy on Training data: 0.9415501905972046
- Accuracy score on Test Data: 0.9187817258883249

3 KNN Model:-

- precision recall f1-score support
-
- 0 1.00 1.00 1.00 85276
- 1 0.91 0.75 0.83 167
-
- accuracy 1.00 85443
- macro avg 0.96 0.88 0.91 85443
- weighted avg 1.00 1.00 1.00 85443

High precision (0.9894) indicates a low rate of false positives, which is crucial in fraud detection to minimize the impact on legitimate transactions.
Balanced F1-score (0.9208) suggests a good overall performance in terms of precision and recall.
For Fraud Detection: Considering the nature of fraud detection where correctly identifying fraudulent transactions (high recall) while maintaining a low false p ositive rate (high precision) is crucial, the SVM model's metrics suggest it is per forming better overall.