Fraud Detection in Credit Card Transactions

15/06/2023

Arun.M

Kuttalam,609801.

8489623143.

Table of Contents

Abstract

In this project Credit card frauds are easy and friendly targets. E-commerce and many other online sites have increased the online payment modes, increasing the risk for online frauds. Increase in fraud rates, researchers started using different machine learning methods to detect and analyze frauds in online transactions. Credit card fraud detection algorithm using machine learning techniques in python. With a lot of people, banks and online retailers being victims of credit card fraud, a model detecting whether the transaction is fraud or not can help in saving a huge amount of money. We used four different types of supervised and unsupervised algorithms to detect the fraud on credit cards payments, This project aims to focus mainly on machine learning algorithms. The algorithms used are 1.Logistic algorithm, 2.Support vector machine algorithm, 3.Naive bayes algorithm, 4.Random Forest alogrithms, The results of the Four algorithms are based on accuracy, precision and recall. The ROC curve is plotted based on the confusion matrix. Main objectives are compared and the algorithm that has the greatest accuracy is considered as the best algorithm that is used to detect the fraud.

Keywords : credit card, fraud detection, Machine learning algorithms.

Introduction

In the last decade, there has been an exponential growth of the Internet. This has sparked the proliferation and increase in the use of services such as e-commerce, tap and pay systems, online bills payment systems etc. As a consequence, fraudsters have also increased activities to attack transactions that are made using credit cards.

In this machine learning project, we solve the problem of detecting credit card fraud transactions using machine numpy, scikit learn, and a few other python libraries. We overcome the problem by creating a binary classifier and experimenting with various machine learning techniques to see which fits better.

We used four different types of supervised and unsupervised algorithms to detect the fraud on credit cards payments, This project aims to focus mainly on machine learning algorithms.

Some familiar Algorithm are used in the project,

1. Logistic Regression.
2. Support Vector Machine(SVM).
3. Naive Bayes.
4. Random Forest.

The increase of people using credit cards in their daily lives, credit card companies should take special care in the security and safety of their customers. According to (Credit card statistics 2021) the number of people using credit cards around the world was 2.8 billion in 2019, in addition70%of those users own a single card at least.

Reports of Credit card fraud in the US rose by 44.7% from 271,927 in 2019 to 393,207 reports in 2020. There are two kinds of credit card fraud, the first one is by having a credit card account opened under your name by an identity thief, reports of this fraudulent behavior increased 48% from 2019 to 2020.

The second type is when an identity thief uses an existing account that you created, and it's usually done by stealing the information of the credit card, reports on this type of fraud increased 9% from 2019 to 2020 (Daly, 2021).

Methodology

1.Data Collection:

- ➢ Gather credit card transaction data from relevant sources, such as financial institutions or credit card processors. Ensure that the dataset includes both legitimate and fraudulent transactions.
- ➢ Data set has taken from kaggle dataset https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

2.Data Preprocessing:

- ➢ Perform necessary data preprocessing steps to prepare the dataset for model training.
- ➢ Removing duplicates or redundant data.
- ➢ Handling missing values by imputation or removal.
- ➢ Scaling numerical features to a similar range (e.g., using normalization or standardization).
- ➢ Encoding categorical variables (e.g., one-hot encoding or label encoding).

3.Dataset Splitting:

- ➢ Split the preprocessed dataset into training, validation, and test sets. The typical split ratios are 70-15-15, but this can vary depending on the dataset size and requirements.

4.Model Selection:

- ➢ Choose appropriate machine learning algorithms for credit card fraud detection.
- ➢ Commonly used algorithms include:
- ➢ **Supervised Learning:**
  1. Logistic Regression.

- ❖ Logistic Regression is basically effective for binary classification problems.

2. Support Vector Machine(SVM).
    - ❖ Support Vector Machine is effective for separating classes in high-dimensional spaces.
3. Naive Bayes.
    - ❖ Naive Bayes algorithm, which is used for classification tasks, like text classification.
4. Random Forests.
    - ❖ Random Forest algorithm is an ensemble of decision trees for improved performance and robustness.

5. Model Training.

- ➢ Train the selected machine learning models using the training dataset.
- ➢ Configure the models with appropriate hyperparameters. Perform hyperparameter tuning using techniques like grid search, random search, or Bayesian optimization to optimize model performance.

6.Model Evaluation:

- ➢ Evaluate the trained models using the validation dataset.
- ➢ Calculate evaluation metrics such as accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC) to assess model performance.
- ➢ Choose the model(s) with the best performance for further evaluation and deployment.

7.Model Deployment:

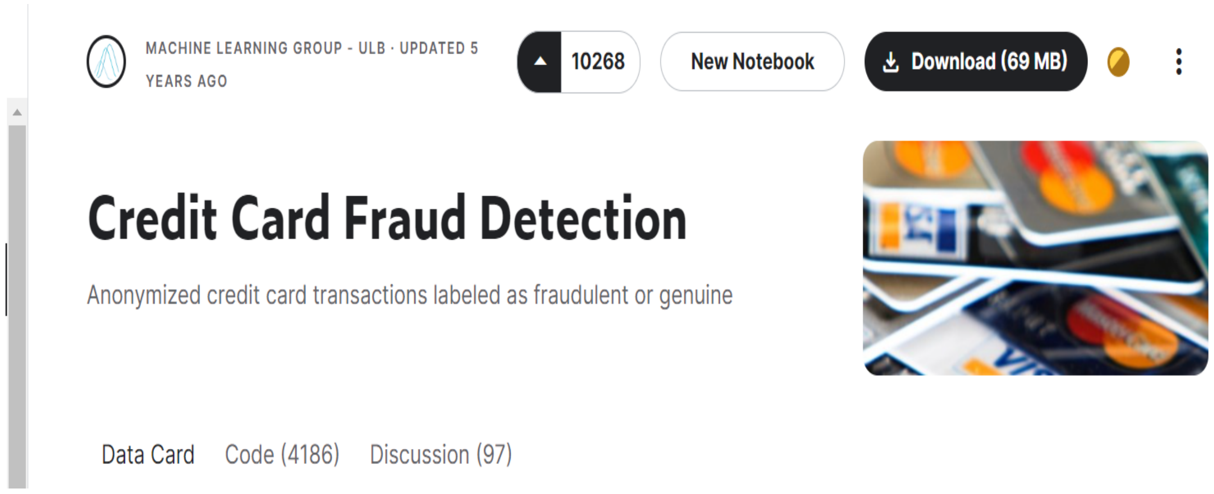
- ➢ The final phase will show evaluations of the models by presenting their efficiency, the accuracies of the models will be presented in addition to any comment observed, to find the best and most suited model for detecting the fraud transactions made by credit card.

Dataset

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.

The dataset was retrieved from an open-source website, Kaggle.com. It contains data of transactions that were made in 2013 by credit card users in Europe, in two days only. The dataset consists of **31 attributes, 284,808 rows**. 28 attributes are numeric variables that due to confidentiality and privacy of the customers have been transformed using PCA transformation, the three remaining attributes are "Time" which contains the elapsed seconds between the first and other transactions of each attribute, "Amount" is the amount of each transaction, and the final attribute "Class" which contains binary variables where "1" is a case of fraudulent transaction, and "0" is not as case of fraudulent transaction.

- ➢ Gather credit card transaction data from relevant sources, such as financial institutions or credit card processors. Ensure that the dataset includes both legitimate and fraudulent transactions.
- ➢ Dataset has taken from kaggle dataset https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud


MACHINE LEARNING GROUP - ULB · UPDATED 5 YEARS AGO

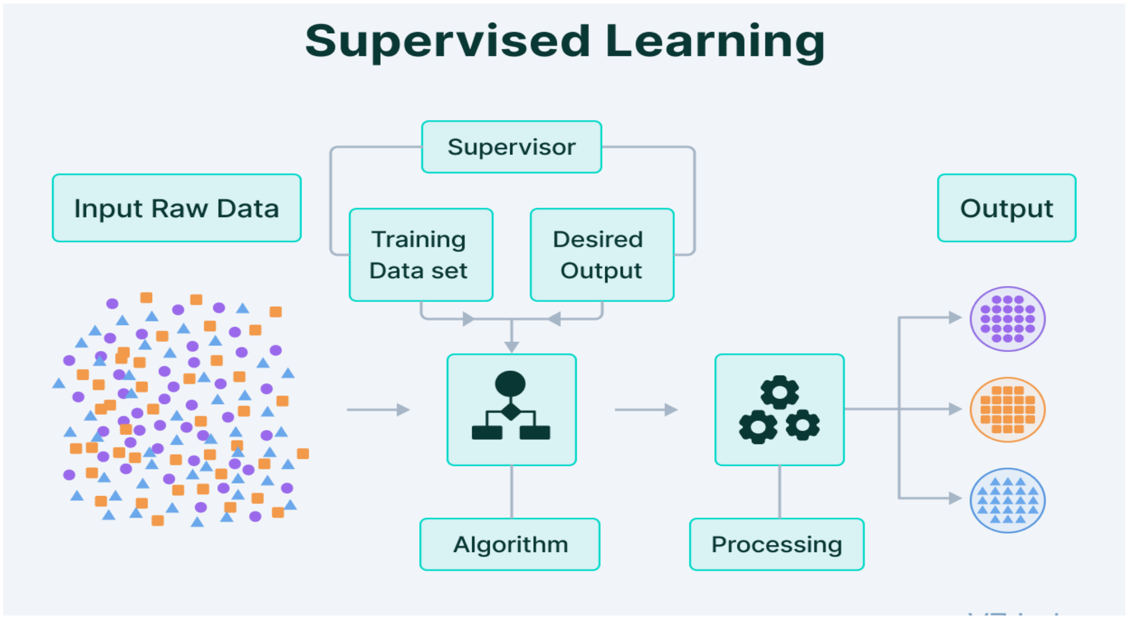
Credit Card Fraud Detection

Anonymized credit card transactions labeled as fraudulent or genuine

ALGORITHMS

<u>**Supervised Machine Learning**</u> **:**

supervised machine learning, is a subcategory of machine learning and artificial intelligence. It is defined by its use of labeled datasets to train algorithms that to classify data or predict outcomes accurately.



Steps Involved in Supervised Learning :

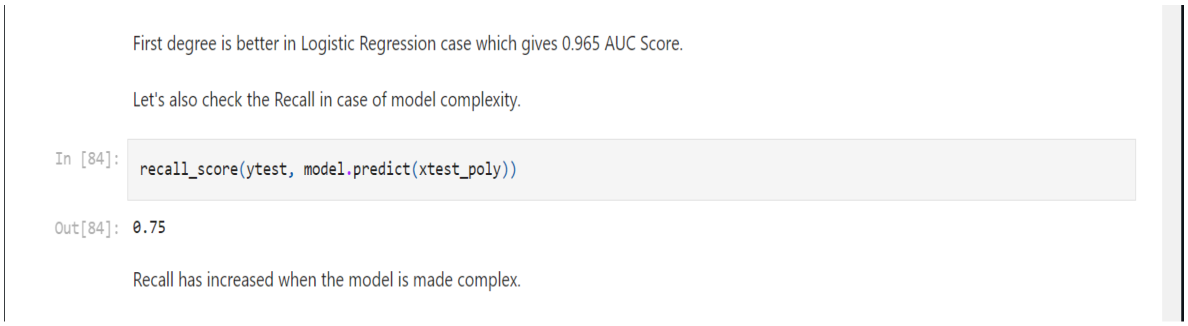
- ➢ First Determine the type of training dataset
- ➢ Collect/Gather the labeled training data.
- ➢ Split the training dataset into training **dataset, test dataset, and validation dataset**.
- ➢ Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.

- ➢ Model predicts the correct output, which means our model is accurateDetermine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- ➢ Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- ➢ Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

(To detect credit card payment fraud detection using supervised machine learning with an imbalanced dataset.)

1.Logistic regression :

- ➢ Logistic Regression is basically effective for binary classification problems.
- ➢ Logistic regression measures the relationship between the categorical target variable and one or more independent variables. It is useful for situations in which the outcome for a target variable can have only two possible types (in other words, it is binary).
- ➢ Logistic Regression is much similar to Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.



First degree is better in Logistic Regression case which gives 0.965 AUC Score.

Let's also check the Recall in case of model complexity.

```
In [84]: recall_score(ytest, model.predict(xtest_poly))

Out[84]: 0.75
```

Recall has increased when the model is made complex.

Fig 3.Logistic regression recall accuracy

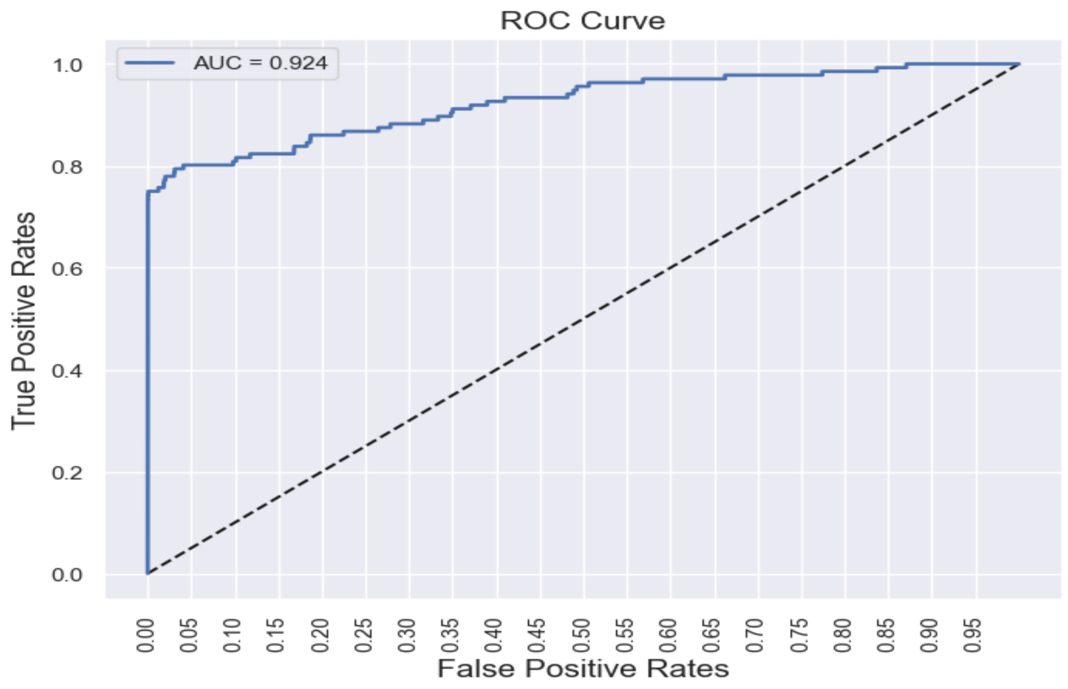


Fig 4.Logistic regression ROC Curve graph

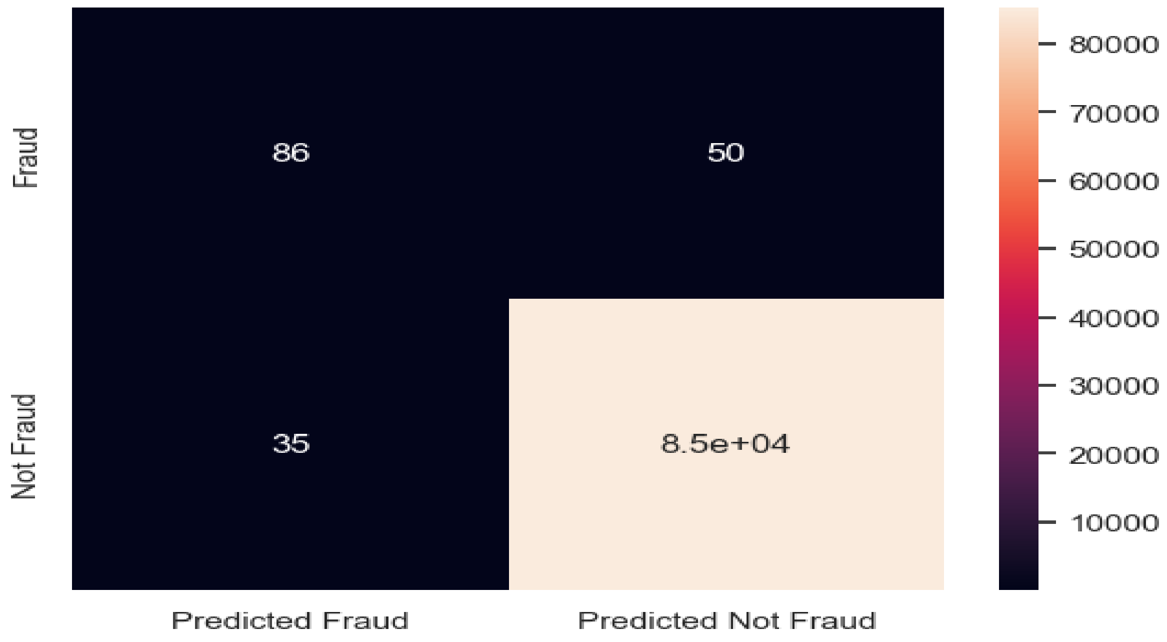
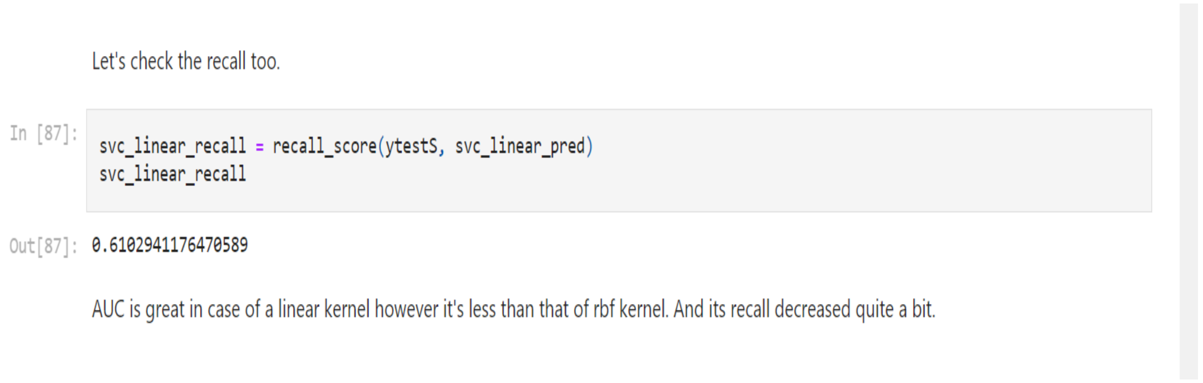


Fig 5.Logistic regression confusion matrix

(Accuracy of Logistic Regression : 0.75)

2.Support Vector Machine (SVM):

- ➢Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- ➢The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- ➢ Support Vector Machine is effective for separating classes in high-dimensional spaces.



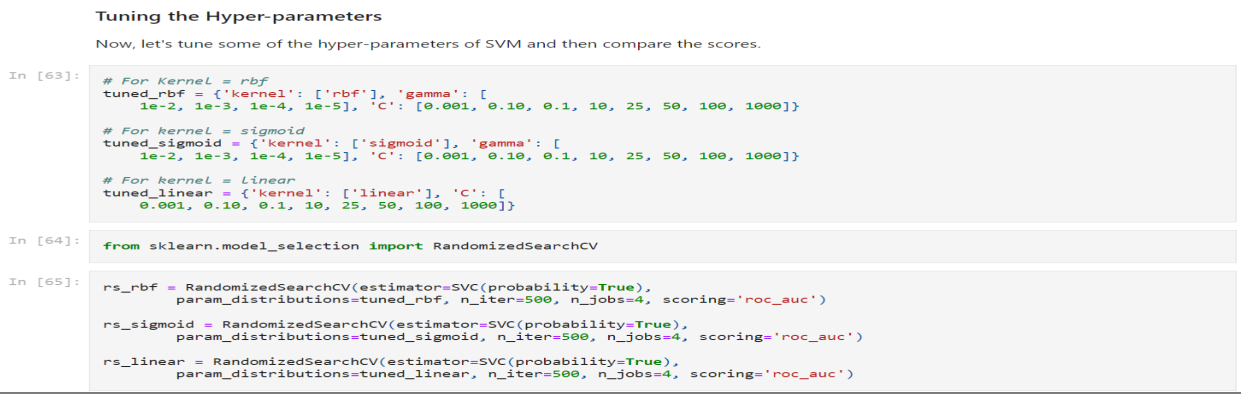
Let's check the recall too.

```
In [87]: svc_linear_recall = recall_score(ytestS, svc_linear_pred)
         svc_linear_recall

Out[87]: 0.6102941176470589
```

AUC is great in case of a linear kernel however it's less than that of rbf kernel. And its recall decreased quite a bit.

Fig 6. SVM recall accuracy



Tuning the Hyper-parameters

Now, let's tune some of the hyper-parameters of SVM and then compare the scores.

```
In [63]: # For Kernel = rbf
         tuned_rbf = {'kernel': ['rbf'], 'gamma': [
             1e-2, 1e-3, 1e-4, 1e-5], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}

         # For kernel = sigmoid
         tuned_sigmoid = {'kernel': ['sigmoid'], 'gamma': [
             1e-2, 1e-3, 1e-4, 1e-5], 'C': [0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}

         # For kernel = Linear
         tuned_linear = {'kernel': ['linear'], 'C': [
             0.001, 0.10, 0.1, 10, 25, 50, 100, 1000]}

In [64]: from sklearn.model_selection import RandomizedSearchCV

In [65]: rs_rbf = RandomizedSearchCV(estimator=SVC(probability=True),
                 param_distributions=tuned_rbf, n_iter=500, n_jobs=4, scoring='roc_auc')

         rs_sigmoid = RandomizedSearchCV(estimator=SVC(probability=True),
                 param_distributions=tuned_sigmoid, n_iter=500, n_jobs=4, scoring='roc_auc')

         rs_linear = RandomizedSearchCV(estimator=SVC(probability=True),
                 param_distributions=tuned_linear, n_iter=500, n_jobs=4, scoring='roc_auc')
```

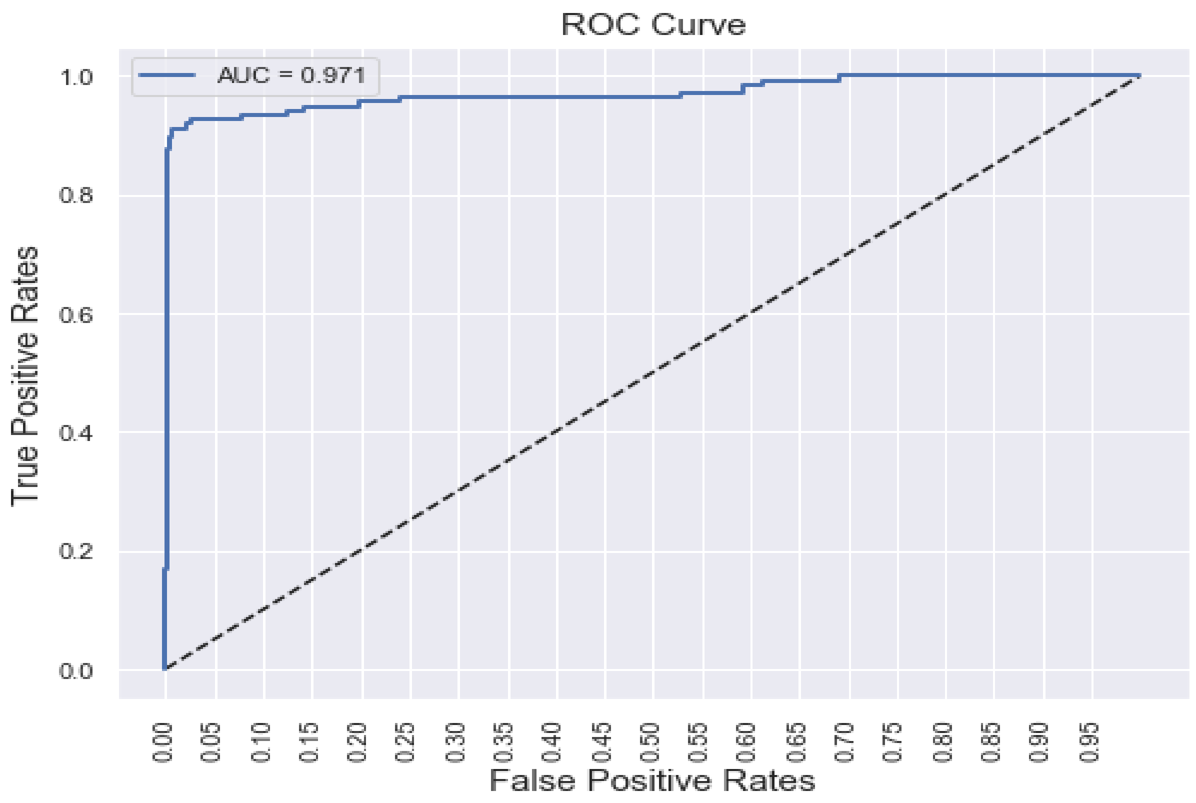


Fig 7.SVM ROC Curve graph

3.Navie Bayes:

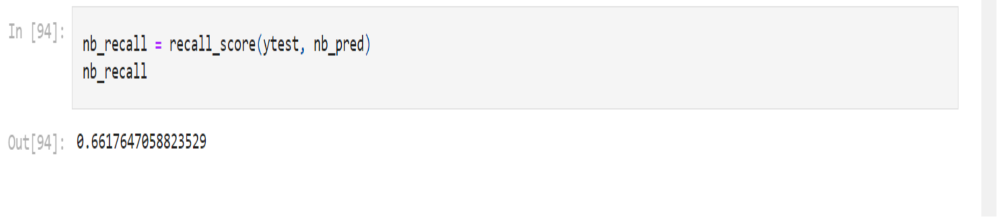
- ➢ Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems.
- ➢ It is mainly used in *text classification* that includes a high-dimensional training dataset.
- ➢ Naive Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

Bayes Theorem.

- ➢ Bayes' theorem is also known as **Bayes' Rule** or **Bayes' law**, which is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability.

➢ The formula for Bayes' theorem is given as:

$$P(A|B)=\frac{P(B|A)P(A)}{P(B)}$$



```
In [94]: nb_recall = recall_score(ytest, nb_pred)
         nb_recall

Out[94]: 0.6617647058823529
```

Fig 8. Naive Bayes recall accuracy

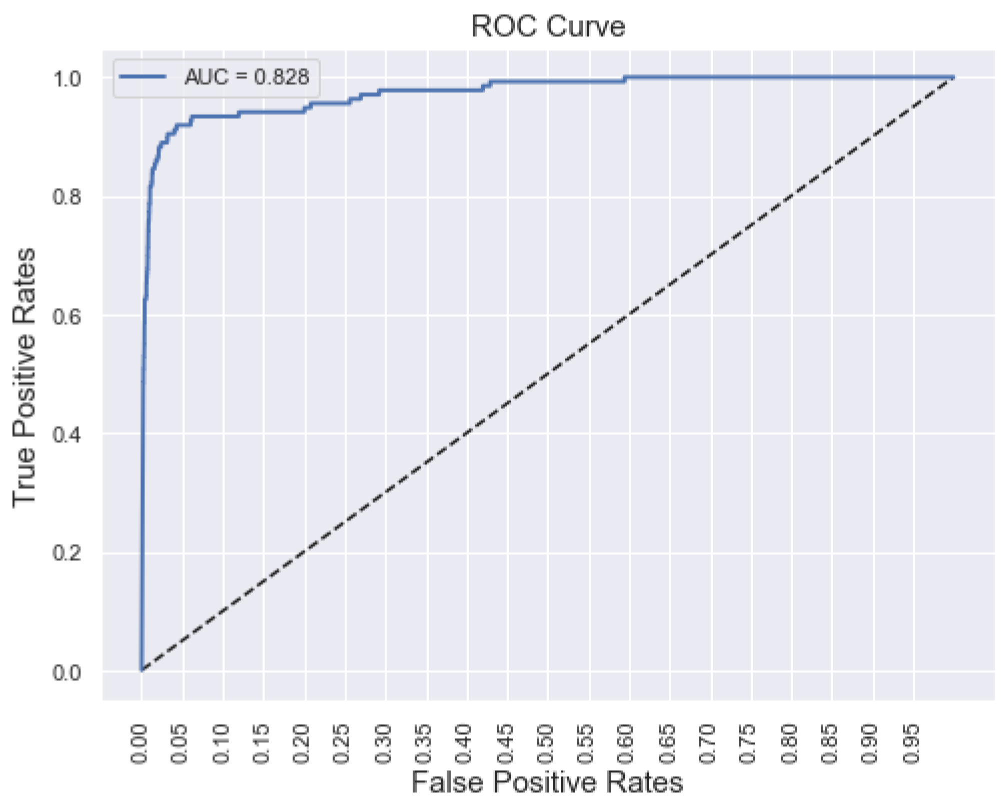


Fig 9.Naive Bayes ROC Curve graph

4.Random Forest :

- ➢ Random Forest algorithm is an ensemble of decision trees for improved performance and robustness.
- ➢ Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- ➢ It takes less training time as compared to other algorithms.
- ➢ It predicts output with high accuracy, even for the large dataset it runs efficiently.
- ➢ It can also maintain accuracy when a large proportion of data is missing.

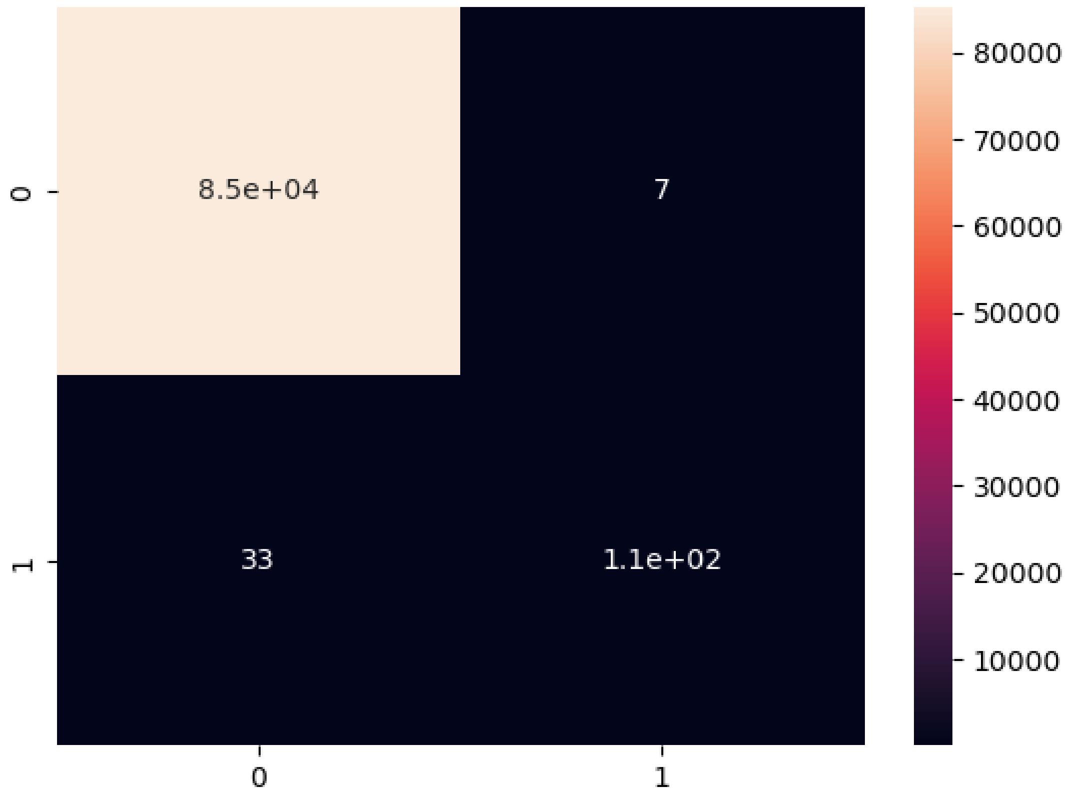
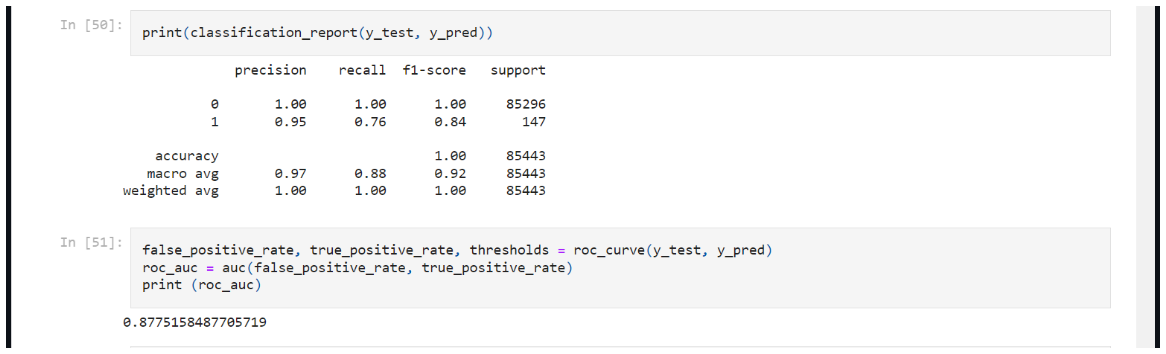


Fig 10.Random Forest Confusion matrix



```
In [50]: print(classification_report(y_test, y_pred))

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     85296
           1       0.95      0.76      0.84       147

    accuracy                           1.00     85443
   macro avg       0.97      0.88      0.92     85443
weighted avg       1.00      1.00      1.00     85443

In [51]: false_positive_rate, true_positive_rate, thresholds = roc_curve(y_test, y_pred)
         roc_auc = auc(false_positive_rate, true_positive_rate)
         print (roc_auc)

0.8775158487705719
```

Fig 11.Random Forest Accuracy.

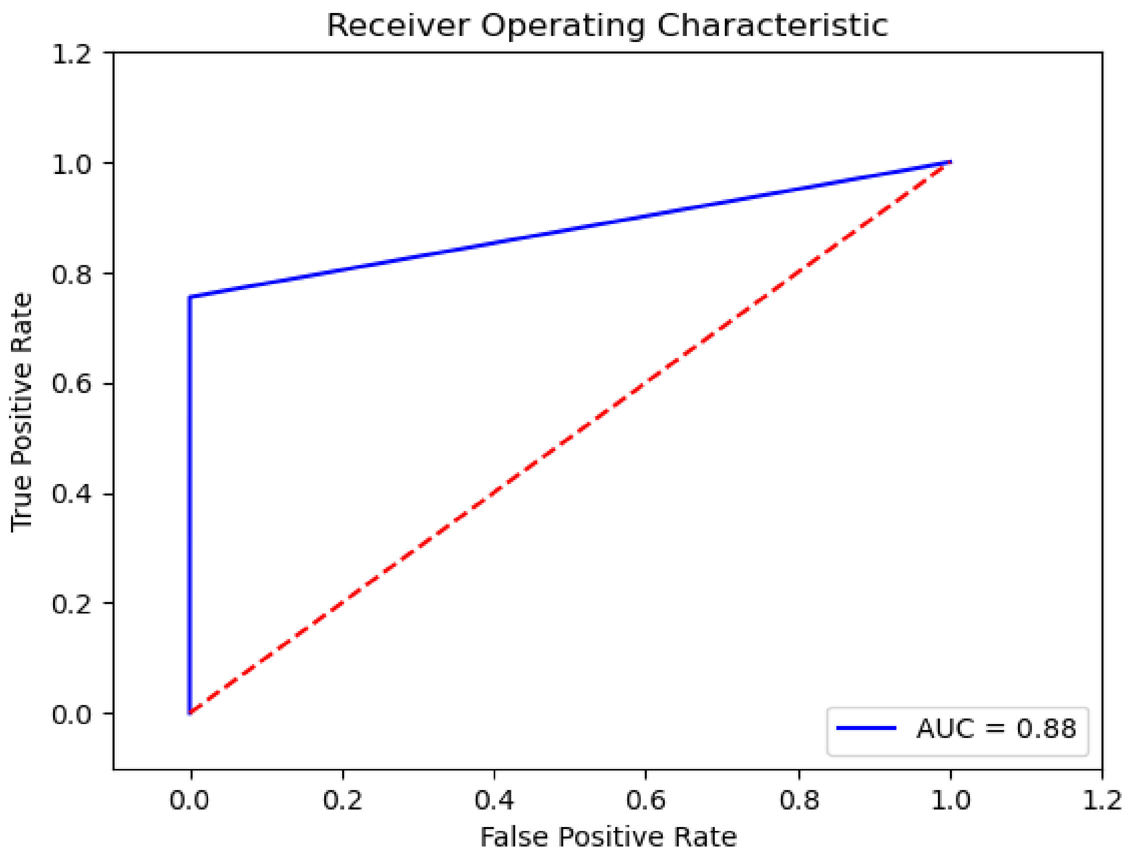
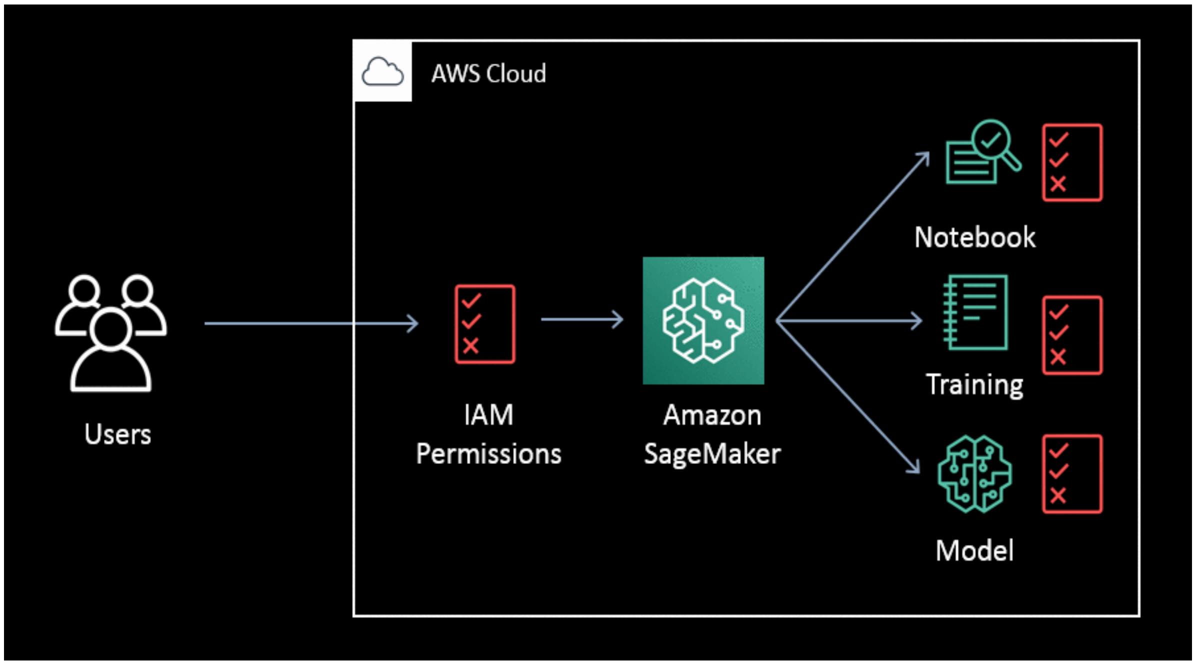


Fig 12. Random Forest AUC graph.

AWS Sage Maker:



➢ Amazon SageMaker is a cloud-based machine-learning platform that helps users create, design, train, tune, and deploy machine learning models in a production-ready hosted environment.
➢ The AWS SageMaker comes with a pool of advantages (know all about it in the next section)

1.BUILD

➢ It provides more than 15 widely used ML Algorithms for training purposes.
➢ It gives the capability to select the required server size for our notebook instance.

- A user can write code (for creating model training jobs) using notebook instances.
- AWS SageMaker helps developers to customize Machine Learning instances with the Jupyter notebook interface.

Test And Tune :

- Set up and import required libraries
- Define a few environment variables and manage them for training the model
- Train and tune the model with Amazon SageMaker
- SageMaker implements hyperparameter tuning by adding a suitable combination of algorithm parameters
- SageMaker uses Amazon S3 to store data as it's safe and secure.

Note: S3 is used for storing and recovering data over the internet.

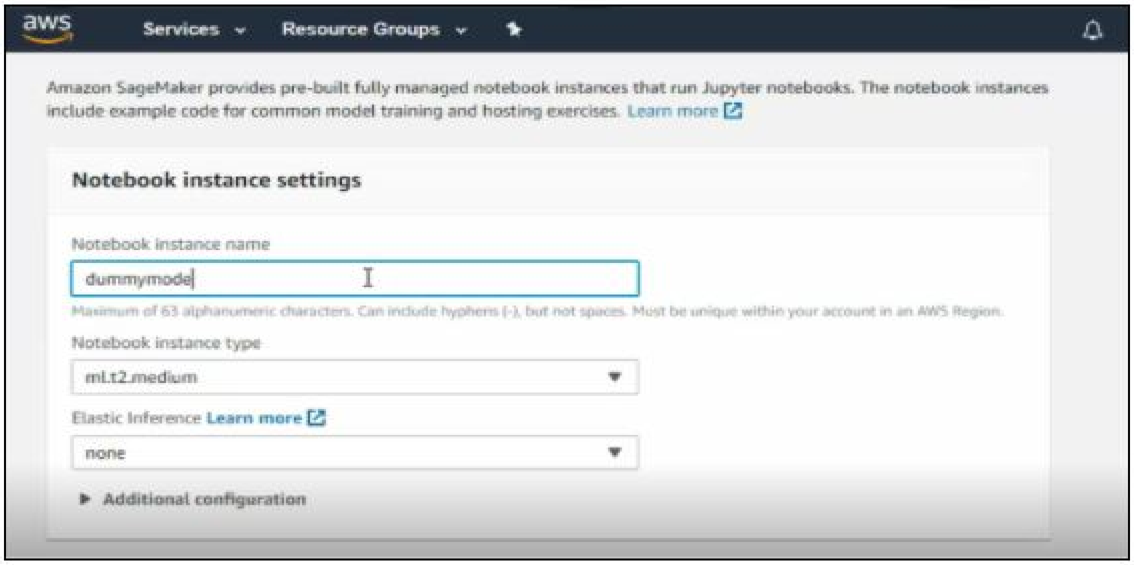
- SageMaker uses ECR for managing Docker containers as it is highly scalable
- SageMaker divides the training data and stores in AWS S3, whereas the training algorithm code is stored in ECR
- Later, SageMaker sets up a cluster for the input data, trains, and stores it in Amazon S3 itself.

Deploy :

- Once tuning is done, models can be deployed to SageMaker endpoints.
- In the endpoints, a real-time prediction is performed.
- Now, evaluate your model and determine whether you have achieved your business goals.

AWS Sage Maker Set-up:

- ➢ To create a notebook instance, use either the SageMaker console or the CreateNotebookInstance API
- ➢ First, open the SageMaker console at https://console.aws.amazon.com/sagemaker/.
- ➢ Once the instance is opened, select Notebook instances -> Create notebook instances. This will create the notebook instance successfully
- ➢ On the instance page, enter the following information:
- ➢ In the Notebook instance name and tab, type a suitable name and tag for your notebook instance.



Amazon SageMaker provides pre-built fully managed notebook instances that run Jupyter notebooks. The notebook instances include example code for common model training and hosting exercises. Learn more

Notebook instance settings

Notebook instance name: dummymode

Maximum of 63 alphanumeric characters. Can include hyphens (-), but not spaces. Must be unique within your account in an AWS Region.

Notebook instance type: ml.t2.medium

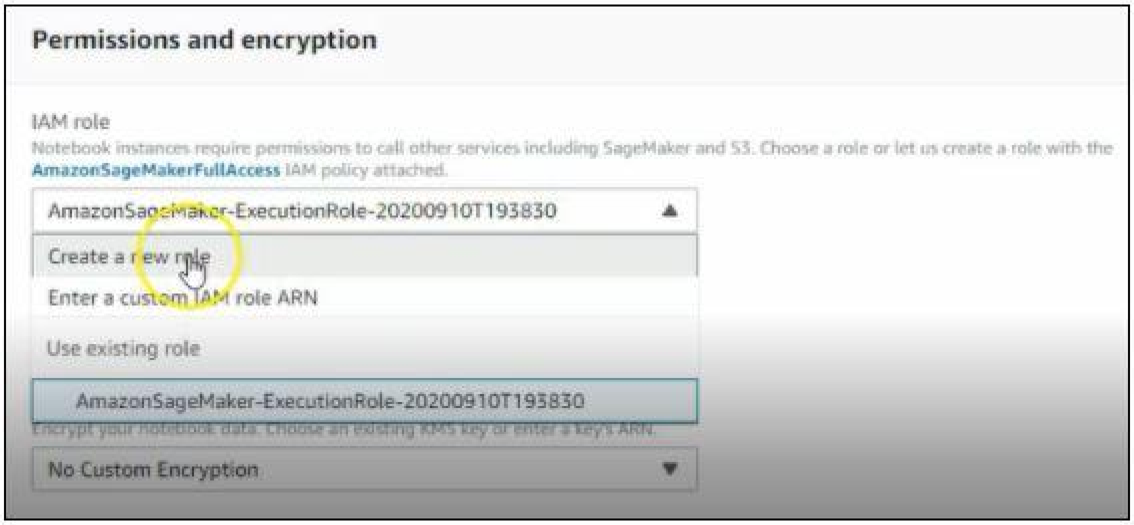
Elastic Inference Learn more: none

▸ Additional configuration

Next, in the Notebook instance type, select an appropriate instance size for your project.

- ➢ In the Elastic Inference option, choose none if you want to skip that option, otherwise, select inference accelerator type in case you are planning to conduct inferences
- ➢ (Optional) In this configuration option, you can specify ML storage volume in MB for notebook instances. Next, specify the IAM role

for the SageMaker model. You can either select an existing IAM role in your account or Create a new role.



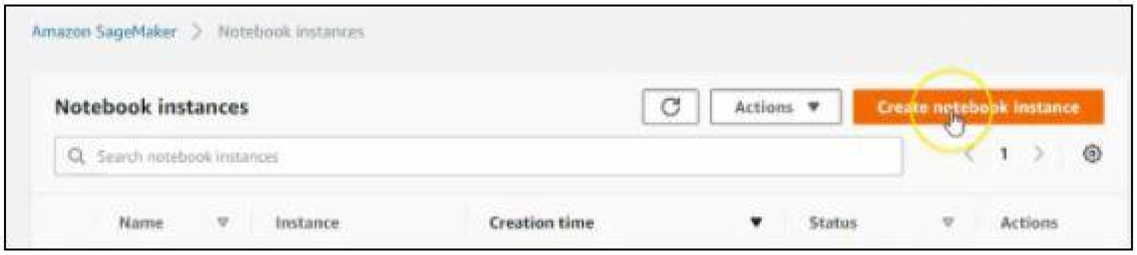
Permissions and encryption

IAM role

Notebook instances require permissions to call other services including SageMaker and S3. Choose a role or let us create a role with the AmazonSageMakerFullAccess IAM policy attached.

AmazonSageMaker-ExecutionRole-20200910T193830
Create a new role
Enter a custom IAM role ARN
Use existing role
AmazonSageMaker-ExecutionRole-20200910T193830
Encrypt your notebook data. Choose an existing KMS key or enter a key's ARN.
No Custom Encryption

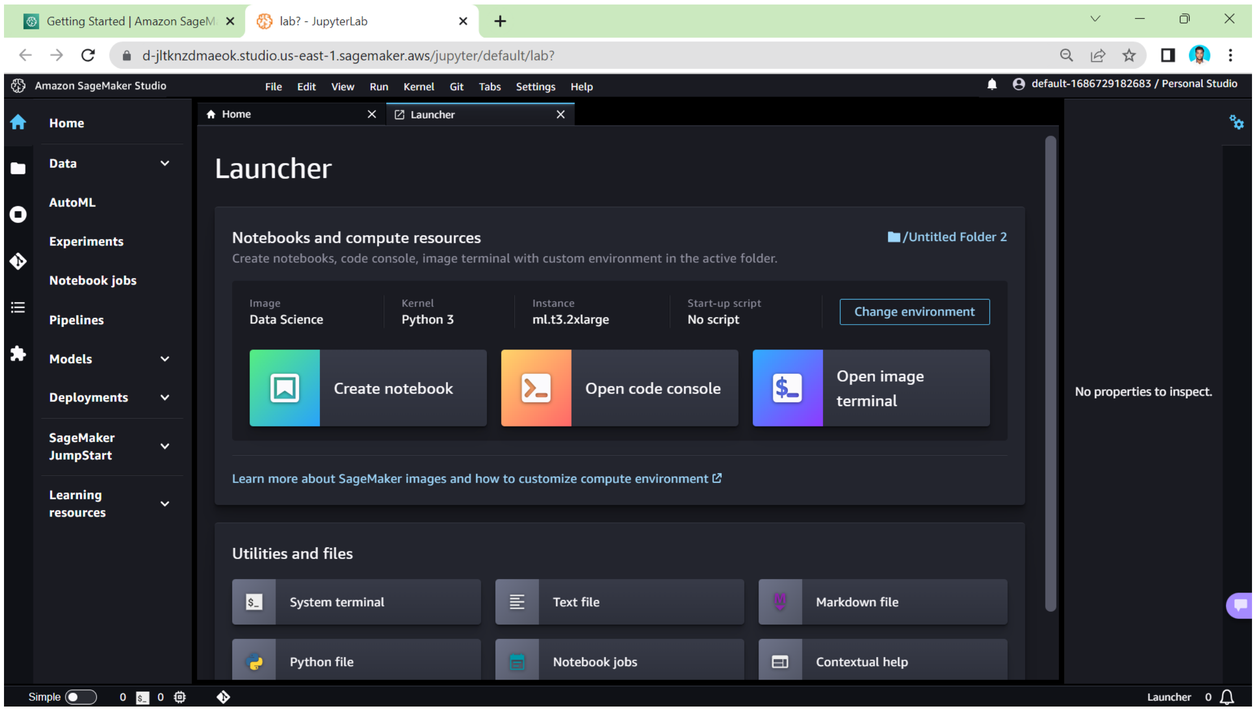
➢ Now, enable root access for all notebook instances. For this, select Enable. In case you want to disable root access, select Disable.
➢ Finally, click on the Create notebook instance.



➢ Within a few minutes, SageMaker creates a Machine Learning Notebook instance and attaches a storage volume.

Create an Amazon SageMaker Notebook Instance :

An Amazon SageMaker notebook instance is a fully managed machine learning (ML) Amazon Elastic Compute Cloud (Amazon EC2) compute instance that runs the Jupyter Notebook App. You use the notebook instance to create and manage Jupyter notebooks for preprocessing data and to train and deploy machine learning models.



Create notebooks, code console, image terminal with custom environment in the active folder.

Image: Data Science
Kernel: Python 3
Instance: ml.t3.2xlarge
Start-up script: No script

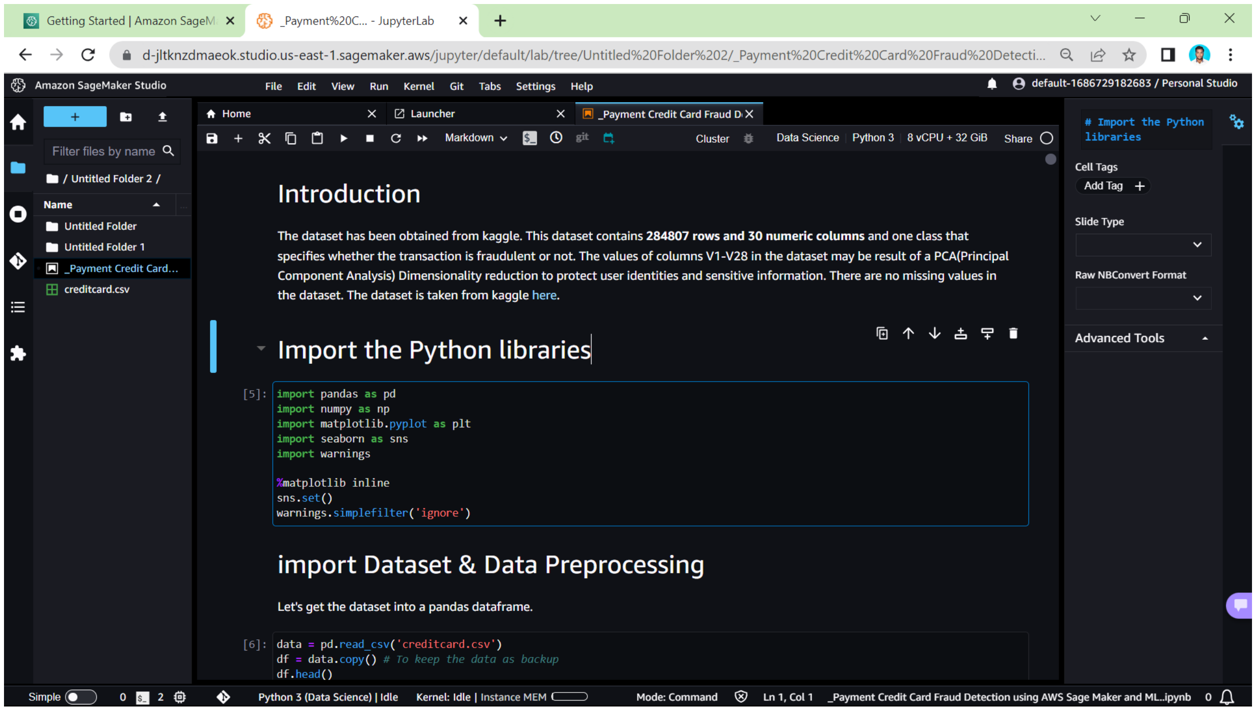
Learn more about SageMaker images and how to customize compute environment

No properties to inspect.

1. Open the Amazon SageMaker console at https://console.aws.amazon.com/sagemaker/.
2. Choose Notebook instances, and then choose Create notebook instances.
3. On the Create notebook instance page, provide the following information (if a field is not mentioned, leave the default values):
    a. For Notebook instance name, type a name for your notebook instance.
    b. For Notebook Instance type, choose ml.t2.medium. This is the least expensive instance type that notebook instances support, and it suffices for this exercise. If a ml.t2.medium instance type

isn't available in your current AWS Region, choose ml.t3.medium.

4. For Platform Identifier, choose a platform type to create the notebook instance on. This platform type dictates the Operating System and the JupyterLab version that your notebook instance is created with. For information about platform identifier type, see Amazon Linux 2 vs Amazon Linux notebook instances. For information about JupyterLab versions, see JupyterLab versioning.
5. Choose Create notebook instance.
   In a few minutes, SageMaker launches an ML compute instance—in this case, a notebook instance—and attaches a 5 GB of Amazon EBS storage volume to it. The notebook instance has a preconfigured Jupyter notebook server, SageMaker and AWS SDK libraries, and a set of Anaconda libraries.
   For more information about creating a SageMaker notebook instance, see Create a Notebook Instance.
6. You tube link for set up sage maker notebook : https://youtu.be/X5CLunIzj3U


Getting Started | Amazon SageM
_Payment%20C... - JupyterLab
d-jltknzdmaeok.studio.us-east-1.sagemaker.aws/jupyter/default/lab/tree/Untitled%20Folder%202/_Payment%20Credit%20Card%20Fraud%20Detecti...
Amazon SageMaker Studio
File Edit View Run Kernel Git Tabs Settings Help
default-1686729182683 / Personal Studio
Filter files by name
/ Untitled Folder 2 /
Name
Untitled Folder
Untitled Folder 1
_Payment Credit Card...
creditcard.csv
Home
Launcher
_Payment Credit Card Fraud D
Markdown
Cluster
Data Science | Python 3 | 8 vCPU + 32 GiB
Share
# Import the Python libraries
Cell Tags
Add Tag
Slide Type
Raw NBConvert Format
Advanced Tools
Simple 0 2
Python 3 (Data Science) | Idle
Kernel: Idle | Instance MEM
Mode: Command
Ln 1, Col 1
_Payment Credit Card Fraud Detection using AWS Sage Maker and ML..ipynb 0

Introduction

The dataset has been obtained from kaggle. This dataset contains **284807 rows and 30 numeric columns** and one class that specifies whether the transaction is fraudulent or not. The values of columns V1-V28 in the dataset may be result of a PCA(Principal Component Analysis) Dimensionality reduction to protect user identities and sensitive information. There are no missing values in the dataset. The dataset is taken from kaggle here.

Import the Python libraries

```
[5]: import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     import seaborn as sns
     import warnings

     %matplotlib inline
     sns.set()
     warnings.simplefilter('ignore')
```

import Dataset & Data Preprocessing

Let's get the dataset into a pandas dataframe.

```
[6]: data = pd.read_csv('creditcard.csv')
     df = data.copy() # To keep the data as backup
     df.head()
```

Accuracy Comparison :

Accuracy Comparison of Four Algorithms

Algorithm	Accuracy Recall Rate
1.Logistic Regression	0.7526470588235294
2.Support Vector Machine(SVM)	0.6102941176470589
3.Naive Bayes	0.6617647058823529
4.Random Forest	0.8775158487705719

(According to our implementation of four supervised learning, Random Forest has a better accuracy rate then the other three algorithms.)

Hardware/Software Requirements :

Hardware Requirements

Processors	Intel Core I5 or AMD R5
RAM	8 GB/16 GB
Disk Space	500 GB upto 1TB

Software Requirements

Server Side	AWS Sage Maker
IDE	Jupyter Notebook
Kernel Type	Python 3
Instance Type	Ml.M5.4xlarge 16 vcpu + 64gib
Operating System	Windows 11

Reference :

➢Dataset has taken from kaggle data collection anonymized credit card transactions labeled as fraudulent or genuine.

➢Dataset link : https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

➢Source Code has referred from- Hamza Nasir- from github

➢**GitHub Link :** Credit Card Fraud Detection using Machine Learning inPython.

https://github.com/hamzanasirr/Credit-Card-Fraud-Detection-using-Machine-Learning

[1] Jiang, Changjun et al. "Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism." IEEE Internet of Things Journal 5 (2018): 3637-3647.

[2]Pumsirirat, A. and Yan, L. (2018). Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann

Machine. International Journal of Advanced Computer Science and Applications, 9(1).

[3]Mohammed, Emad, and Behrouz Far. “Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study.” IEEE Annals of the History of Computing, IEEE, 1 July 2018, doi.ieeecomputersociety.org/10.1109/IRI.2018.00025.

Conclusion :

In this paper we developed a novel method for fraud detection, where customers are grouped based on their transactions and extract behavioral patterns to develop a profile for every cardholder. Then different classifiers are applied on three different groups; later rating scores are generated for every type of classifier.. We finally observed that Logistic regression and random forest are the algorithms that gave better results.