Online Fraud Detection System

by

NIKHIL CHORDIA A	**21BLC1645**
SUJAY GHOSH	**21BLC1607**
HEMANT M MEHTA	**21BLC1379**

A project report submitted to

Dr. KALAIVANAN K

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfillment of the requirements for the course of

BCSE308L- COMPUTER NETWORKS

in

B. Tech. ELECTRONICS AND COMPUTER ENGINEERING

Vandalur – Kelambakkam
Road Chennai – 600127
JULY 2023

BONAFIDE CERTIFICATE

Certified that this project report entitled “ONLINE FRAUD DETECTION SYSTEM” is a bonafide work **of NIKHIL CHORDIA A 21BLC1645, SUJAY GHOSH 21BLC1607, HEMANT M MEHTA 21BLC1379** who carried out the Project work under my supervision and guidance for BCSE308L-COMPUTER NETWORKS.

Dr. KALAIVANAN K

Assistant Professor(Sr.)

School of Electronics Engineering (SENSE), VIT University, Chennai

Chennai – 600 127.

ABSTRACT

We lived in the era of technology, where online websites have made communication and interaction much easier than in the early decade. The modern world is changing so quickly, and emerging new trends simplify our lives, but on the other hand, it will also create risks. A website acts as a more admired platform by the user to access different web applications. Because of the functions of the website, all the data on the website is available in the properstructure, which makes the working environment much more convenient and efficient. The websites may undoubtedly be a precious resource for young people, but it may also cause severe problems. It is really easy to build a fake website that looks like an original website where hackers like to deceive consumers and businesses for their own benefit. Researchers highlighted the current situation regarding the usage of online services, as well as the threats that have an impact on users. In addition, researchers have suggested that new laws and regulations should be implemented for preventing cybercrime. This presentation focuses on online fraud detection using machine learning programs.The goal of this presentation is to provide a clear overview of the topic and highlight the importance of using ML for fraud detection in today’s digital world.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. KALAIVANAN K**, Assistant Professor (Sr.), School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Susan Elias**, Dean of the School of Electronics Engineering, VIT Chennai, for extending the facilities of the school towards our project and for her unstinting support.

We express our thanks to our Head of the Department **Dr. Annis Fathima A** for her support throughout the course of this project.

We also take this opportunity to thank all the faculty of the school for their support and the wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the tough times of this pandemic throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

NIKHIL CHORDIA A   SUJAY GHOSH   HEMANT MEHTA

TABLE OF CONTENTS

Chapter-1

INTRODUCTION

1.1 OBJECTIVES AND GOALS

In this project we have tried to implement solutions to handle the problems in online fraud detection. The following are the major objectives we will try to implement. The main objective of this research is to build statistical machine-learning models using supervised learning techniques to detect fraudulent transactions in real-time with great accuracy. To come up with various statistical techniques to deal with the imbalanced nature of the data. Like synthetic scaling of the data, which can decrease the skew without affecting the data integrity. This can help to improve the accuracy of the models. To build an application that can detect the legitimacy of the transaction in real time and increase security to prevent fraud.

1.2 APPLICATIONS OF MACHINE LEARNING

Machine learning programming has a wide range of applications across various industries. Some common applications of ML programming include:

- Image and speech recognition
- Natural language processing

- Fraud detection
- Recommendation systems
- Predictive maintenance
- Autonomous vehicles
- Healthcare diagnosis and treatment
- Financial forecasting
- Energy load forecasting
- Social media sentiment analysis

These are just a few examples of how ML programming is used in differentfields to improve efficiency, accuracy, and decision-making.

1.3 FEATURES

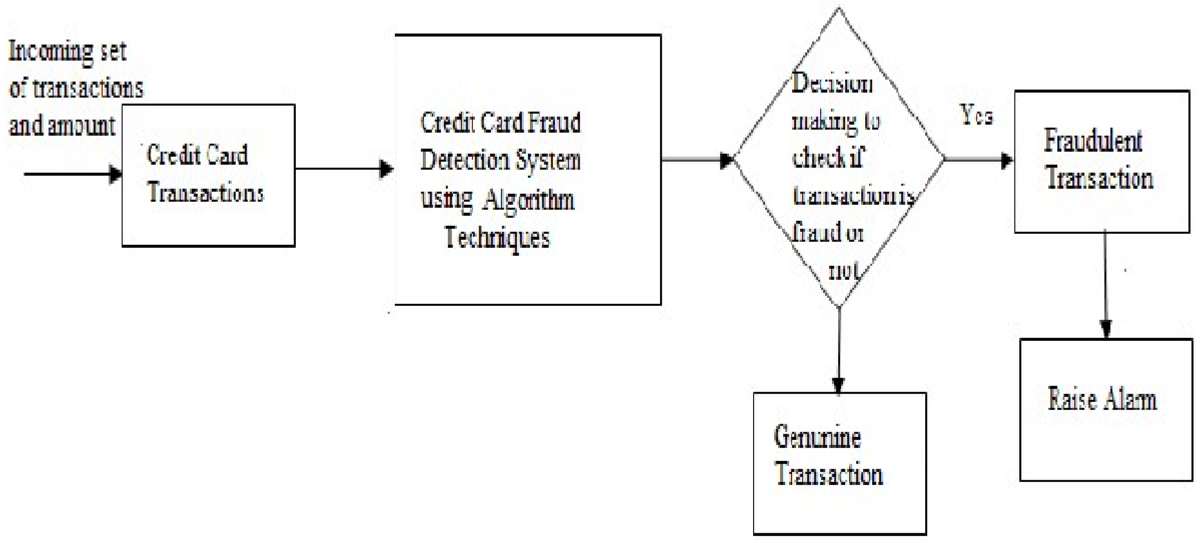
- User behavior analysis: analyzing user behavior patterns to detect any unusual or suspicious activity.
- Anomaly detection: identifying any unusual activity that deviates from normal patterns, such as a sudden increase in transactions or a change in user location.
- Real-time monitoring: continuously monitoring transactions and user behavior in real-time to quickly detect and prevent fraudulent activity.
- Historical data analysis: analyzing historical data to identify patterns and trends that may indicate fraudulent behavior.
- Risk scoring: assigning a risk score to each transaction or user based on the likelihood of fraudulent activity.

- Machine learning algorithms: using various machine learning algorithms such as decision trees, logistic regression, and neural networks to identify patterns and predict fraudulent behavior.

Chapter-2

DESIGN

2.1 BLOCK DIAGRAM:


Incoming set of transactions and amount
Credit Card Transactions
Credit Card Fraud Detection System using Algorithm Techniques
Decision making to check if transaction is fraud or not
Yes
Fraudulent Transaction
Raise Alarm
Genunine Transaction

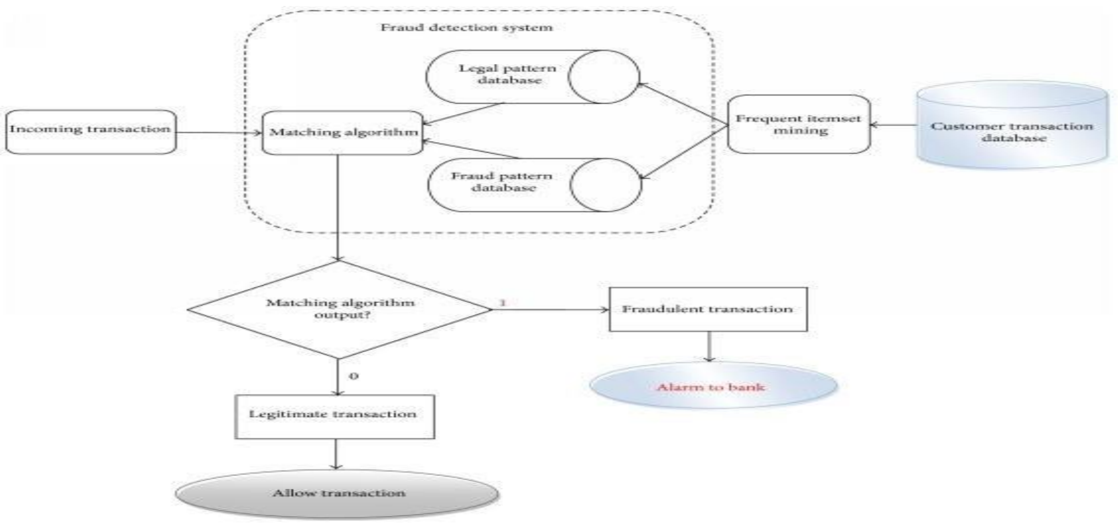


Fig Block diagrams of working of the fraud detection system

In the above block diagrams, we can infer that once a transaction is initiated for example credit card transaction in this case, the machine learning algorithm analysis the transaction i.e. who is the sender, the receiver, the mode of transaction etc and as a outcome gives a result which tells us whether the transaction is a fraudulent one or not. If it a valid and legitimate transaction it allows it or else it states that the intended receiver and transaction is a fraud.

Chapter-3

SOFTWARE - CODING AND ANALYSIS

3.1 SOFTWARE WE HAVE USED:

1) PYTHON: Python is a general-purpose programming language, so it can be used for many things. Python is used for web development, AI, machine learning, operating systems, mobile application development, and video games

2) pandas: pandas is a powerful data manipulation and analysis library. It provides data structures like DataFrames that allow you to work with structured data efficiently.

3) numpy: numpy is a fundamental package for scientific computing with Python. It provides support for multi-dimensional arrays and various mathematical operations, making it useful for numerical computations.

4) matplotlib: matplotlib is a widely used plotting library in Python. It provides a flexible and comprehensive set of functions for creating various types of visualizations, including line plots, scatter plots, histograms, and more.

5) seaborn: seaborn is a data visualization library built on top of matplotlib. It provides a high-level interface for creating visually appealing statistical graphics. It simplifies the creation of complex visualizations and enhances the default styles of matplotlib.

6) sklearn.svm: sklearn.svm is a module from scikit-learn, a popular machine learning library. It provides support vector machine (SVM) algorithms for classification and regression tasks.

7) sklearn.ensemble: sklearn.ensemble is another module from scikit-learn that contains ensemble learning algorithms. It includes methods like random forests, which combine multiple decision trees to make predictions.

8) sklearn.linear_model: sklearn.linear_model is a module from scikit-learn that includes various linear models, such as logistic regression, which is commonly used for binary classification tasks.

9) sklearn.model_selection: sklearn.model_selection provides tools for model selection and evaluation, including train-test splitting, cross-validation, and grid search.

10) sklearn.metrics: sklearn.metrics contains evaluation metrics for assessing model performance. It includes functions for calculating accuracy, precision, recall, F1-score, and more.

11) sklearn.feature_selection: sklearn.feature_selection provides methods for feature selection, which aims to select the most relevant features from a dataset. The SelectKBest and mutual_info_classif functions are used for this purpose.

12) sklearn.covariance: sklearn.covariance includes functions and classes for estimating covariance matrices. In the given code snippet, EllipticEnvelope is used for outlier detection based on the assumption of an elliptical distribution.

13) sklearn.cluster: sklearn.cluster provides methods for clustering, including the K-means algorithm, which is available through the KMeans class.

14) sklearn.preprocessing: sklearn.preprocessing offers various data preprocessing techniques, such as scaling and normalization. StandardScaler and RobustScaler are used for standardization and robust scaling, respectively.

15) sklearn.neighbors: sklearn.neighbors includes algorithms for nearest neighbor-based tasks. In the given code snippet, it is used for kernel density estimation using the KernelDensity class.

16) warnings: The warnings module provides functions for handling warning messages. In this code snippet, the filterwarnings function is used to suppress warning messages.

17) COLAB: it is the platform using for running the code.

3.3 CODE :

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
!pip install seaborn
import seaborn as sns
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report,confusion_matrix
from sklearn.feature_selection import SelectKBest, mutual_info_classif
from sklearn.covariance import EllipticEnvelope
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler, RobustScaler
from sklearn.model_selection import cross_val_score, ShuffleSplit, cross_val_predict
from sklearn.model_selection import GridSearchCV
from sklearn.neighbors import KernelDensity
pd.set_option("display.precision", 2)
scores = ["precision", "recall"]
def print_dataframe(filtered_cv_results):
    """Pretty print for filtered dataframe"""
    for mean_precision, std_precision, mean_recall, std_recall, params in zip(
        filtered_cv_results["mean_test_precision"],
        filtered_cv_results["std_test_precision"],
        filtered_cv_results["mean_test_recall"],
        filtered_cv_results["std_test_recall"],
        filtered_cv_results["params"],
    ):
        print(
            f"precision: {mean_precision:0.3f} (±{std_precision:0.03f}),"
            f" recall: {mean_recall:0.3f} (±{std_recall:0.03f}),"
            f" for {params}"
        )
    print()
def refit_strategy(cv_results):
    scores = ["precision", "recall"]
    # print the info about the grid-search for the different scores
    precision_threshold = 0.96
    while True:
        cv_results_ = pd.DataFrame(cv_results)
        print("All grid-search results:")
        print_dataframe(cv_results_)
        # Filter-out all results below the threshold
        high_precision_cv_results = cv_results_[
            cv_results_["mean_test_precision"] > precision_threshold
        ]
        print(f"Models with a precision higher than {precision_threshold}:")
        print_dataframe(high_precision_cv_results)
        high_precision_cv_results = high_precision_cv_results[
            [
                "mean_score_time",
                "mean_test_recall",
                "std_test_recall",
                "mean_test_precision",
                "std_test_precision",
                "rank_test_recall",
                "rank_test_precision",
                "params",
            ]
        ]
        # Select the most performant models in terms of recall
        # (within 1 sigma from the best)
        best_recall_std = high_precision_cv_results["mean_test_recall"].std()
        best_recall = high_precision_cv_results["mean_test_recall"].max()
        best_recall_threshold = best_recall - best_recall_std

        high_recall_cv_results = high_precision_cv_results[
            high_precision_cv_results["mean_test_recall"] > best_recall_threshold
        ]
        print(
            "Out of the previously selected high precision models, we keep all the\n"
            "the models within one standard deviation of the highest recall model:"
        )
        print_dataframe(high_recall_cv_results)
        try:
            # From the best candidates, select the fastest model to predict
            fastest_top_recall_high_precision_index = high_recall_cv_results[
                "mean_score_time"
            ].idxmin()
        except:
            precision_threshold -= 0.02
        break
    print(
        "\nThe selected final model is the fastest to predict out of the previously\n"
        "selected subset of best models based on precision and recall.\n"
        "Its scoring time is:\n\n"
        f"{high_recall_cv_results.loc[fastest_top_recall_high_precision_index]}"
    )
    return fastest_top_recall_high_precision_index
```

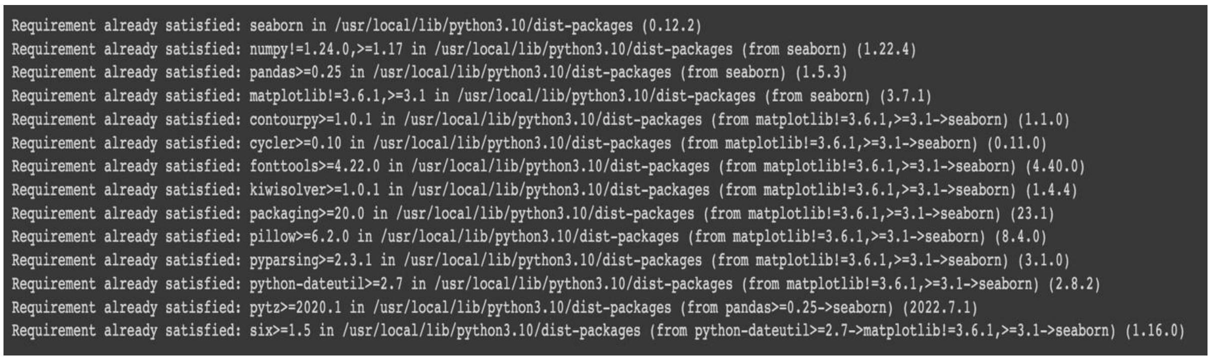
```
import warnings
warnings.filterwarnings('ignore')
_ = '-'
print("\nDataset size - ", df.shape)
print(_*100, end='\n\n')
print(df.info())
print(_*100, end='\n\n')
df = df.drop(columns='Time')
print('I\'ll drop the Time column as credit card default may not be associated with Time. Now
we have 30 features.\n')
print(_*100, end='\n\n')
print("Lets check if there are any NULLs or NA in our dataset. Number of NA values -
",df.isna().sum().sum())
print('\nSince we don\'t have any NAs we\'ll proceed further.\n')
print(_*100, end='\n\n')
print("Also, I want to check if there are any duplicates in the dataset. Duplicates can affect
our analysis.")
print("\nNumber of Duplicate rows in the data set - ", df.duplicated().sum())
df = df.drop_duplicates()
print("\nNow that we have identified the presence of duplicates in our data, I will remove the
redundant rows.\n\nDataset Final Size - ", df.shape)
print(_*100, end='\n\n')
class_freq = df['Class'].value_counts()
print("Overview of distribution of Fraud and non-fraud entries in Data.\n")
print(class_freq)
print("\n% Fraud cases in the dataset - ", class_freq[1]/class_freq[0])
print(_*100, end='\n\n')
print("Let us have a look at the descriptive statistics for this data.")
df.describe(percentiles=[.01, .99])
feature_selector = SelectKBest(mutual_info_classif)
feature_selected = feature_selector.fit_transform(df.drop(columns='Class'), df['Class'])
features = feature_selector.get_feature_names_out()
df = df.sample(frac=1, random_state=38)
frauds = df.loc[df['Class'] == 1]
no_frauds = df.loc[df['Class']==0][:frauds.shape[0]]
norm_distributed_df = pd.concat([frauds, no_frauds], axis = 0)
new_df = norm_distributed_df.sample(frac=1, random_state = 38)
new_df = new_df.drop_duplicates()
new_df.shape
X = new_df.loc[:, features]
y = new_df.loc[:, 'Class']
ss = StandardScaler()
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=39)
print(_*100, end='\n\n')
print("Splitting the dataset into training and test data we now have - ")
print("\nTrain Size: %s" %len(y_train))
print("Test Size: %s" %len(y_test))
print(_*150, end='\n\n')
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
```

```
print("I am using Elliptic Envelope to determine outliers. However, \
it may not be the best choice here as it assumes that the underlying data is Gaussian. \
I am assuming that 10% data instances might be outliers so using contamination as 0.1 in
Elliptic Envelope\n")
print(_*100, end='\n\n')
outlier_detector = EllipticEnvelope(contamination=.05)
outlier_detector.fit(X_train)
indices = outlier_detector.predict(X_train)
bool_indices = np.where(indices == 1, True, False)
X_train = X_train[bool_indices]
y_train = y_train[bool_indices]
print("After Removing the outliers we have %s data instances" %len(X_train))
km = KMeans(n_clusters=2, n_init= 10, random_state=39).fit(X)
cluster = km.predict(X)
print(classification_report(cluster, y))
tc = km.fit_transform(X)
sns.scatterplot(data=tc, x=tc[:,0], y=tc[:,1], hue=y)
tuned_parameters = [
    {"kernel": ["rbf"], "gamma": [1e-3, 1e-4], "C": [1, 10, 100, 1000]},
    {"kernel": ["linear"], "C": [1, 10, 100, 1000]},
]
grid_search = GridSearchCV(
    SVC(random_state = 53), tuned_parameters, scoring=scores, refit=refit_strategy
)
grid_search.fit(X_train, y_train)
print(_*100, end='\n\n')
print("\n\nBest Parameters in our Grid Search")
print(grid_search.best_params_)
print(_*100, end='\n\n')
y_pred = grid_search.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
tuned_parameters = [
    {"max_depth": [4,6,8]}
]
grid_search = GridSearchCV(
    RandomForestClassifier(max_samples=0.8, random_state=53), tuned_parameters,
scoring=scores, refit=refit_strategy
)
grid_search.fit(X_train, y_train)
print(_*100, end='\n\n')
print("\n\nBest Parameters in our Grid Search")
print(grid_search.best_params_)
print(_*100, end='\n\n')
y_pred = grid_search.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
params = {"bandwidth": np.logspace(-1, 1, 20)}
grid = GridSearchCV(KernelDensity(), params)
grid.fit(df.loc[df['Class']==1])
```

```
print("Best bandwidth: {0}".format(grid.best_estimator_.bandwidth))
print(_*100, end='\n\n')
# use the best estimator to compute the kernel density estimate
kde = grid.best_estimator_
latest_data = kde.sample(1000, random_state=10)
latest_df = pd.DataFrame(latest_data, columns=df.columns)
latest_df['Class'] = 1
latest_df = latest_df[latest_df['Amount']>0]
oversample = pd.concat([latest_df, df], axis =0)
oversample = oversample.sample(frac=1, random_state=38)
frauds = oversample.loc[oversample['Class'] == 1]
no_frauds = oversample.loc[oversample['Class']==0][:frauds.shape[0]]
norm_distributed_df = pd.concat([frauds, no_frauds], axis = 0)
oversample_new_df = norm_distributed_df.sample(frac=1, random_state = 38)
print('Shape of the new dataset is ', oversample_new_df.shape)
print(_*100, end='\n\n')
feature_selector = SelectKBest(mutual_info_classif)
feature_selected=feature_selector.fit_transform(oversample_new_df.drop(columns='Class'),
oversample_new_df['Class'])
features = feature_selector.get_feature_names_out()
print('Features selected with Mutual Information are ',features)
print(_*100, end='\n\n')
X = oversample_new_df.loc[:, features]
y = oversample_new_df.loc[:, 'Class']
ss = StandardScaler()
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=39)
print("Splitting the dataset into training and test data we now have - ")
print("\nTrain Size: %s", len(y_train))
print("Test Size: %s", len(y_test))
print(_*150, end='\n\n')
X_train = ss.fit_transform(X_train)
X_test = ss.transform(X_test)
print("I am using Elliptic Envelope to determine outliers. However, \it may not be the best
choice here as it assumes that the underlying data is Gaussian. \
I am assuming that 10% data instances might be outliers so using contamination as 0.1 in
Elliptic Envelope")
print(_*100, end='\n\n')
outlier_detector = EllipticEnvelope(contamination=.1)
outlier_detector.fit(X_train)
indices = outlier_detector.predict(X_train)
bool_indices = np.where(indices == 1, True, False)
X_train = X_train[bool_indices]
y_train = y_train[bool_indices]
print('After removing outliers from the training dataset we now have %s training instances'
%(len(y_train)))
tuned_parameters = [
    {"kernel": ["rbf"], "gamma": [1e-3, 1e-4], "C": [1, 10, 100, 1000]},
    {"kernel": ["linear"], "C": [1, 10, 100, 1000]},
]
grid_search = GridSearchCV(
    SVC(random_state=53), tuned_parameters, scoring=scores, refit=refit_strategy
)
grid_search.fit(X_train, y_train)
print(_*100, end='\n\n')
print("\n\nBest Parameters in our Grid Search")
print(grid_search.best_params_)
print(_*100, end='\n\n')
y_pred = grid_search.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
tuned_parameters = [
    {"max_depth": [4,6,8]}
]
grid_search = GridSearchCV(
    RandomForestClassifier(max_samples=0.8, random_state=53), tuned_parameters,
scoring=scores, refit=refit_strategy
)
grid_search.fit(X_train, y_train)
print(_*100, end='\n\n')
print("Best Parameters in our Grid Search")
print(grid_search.best_params_)
print(_*100, end='\n\n')
y_pred = grid_search.predict(X_test)
print("\nClassification Report:\n")
print(classification_report(y_test, y_pred))
```

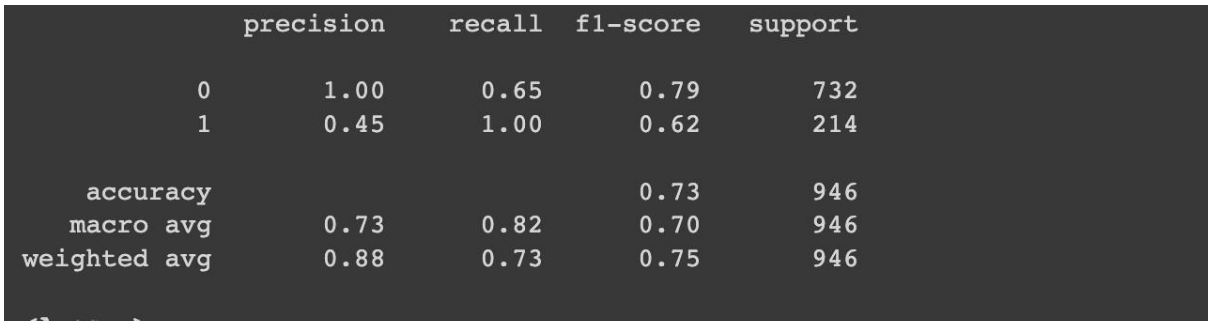
<u>3.5 SNAPSHOTS OF ALL THE SOFTWARES USED :</u>

OUTPUT:



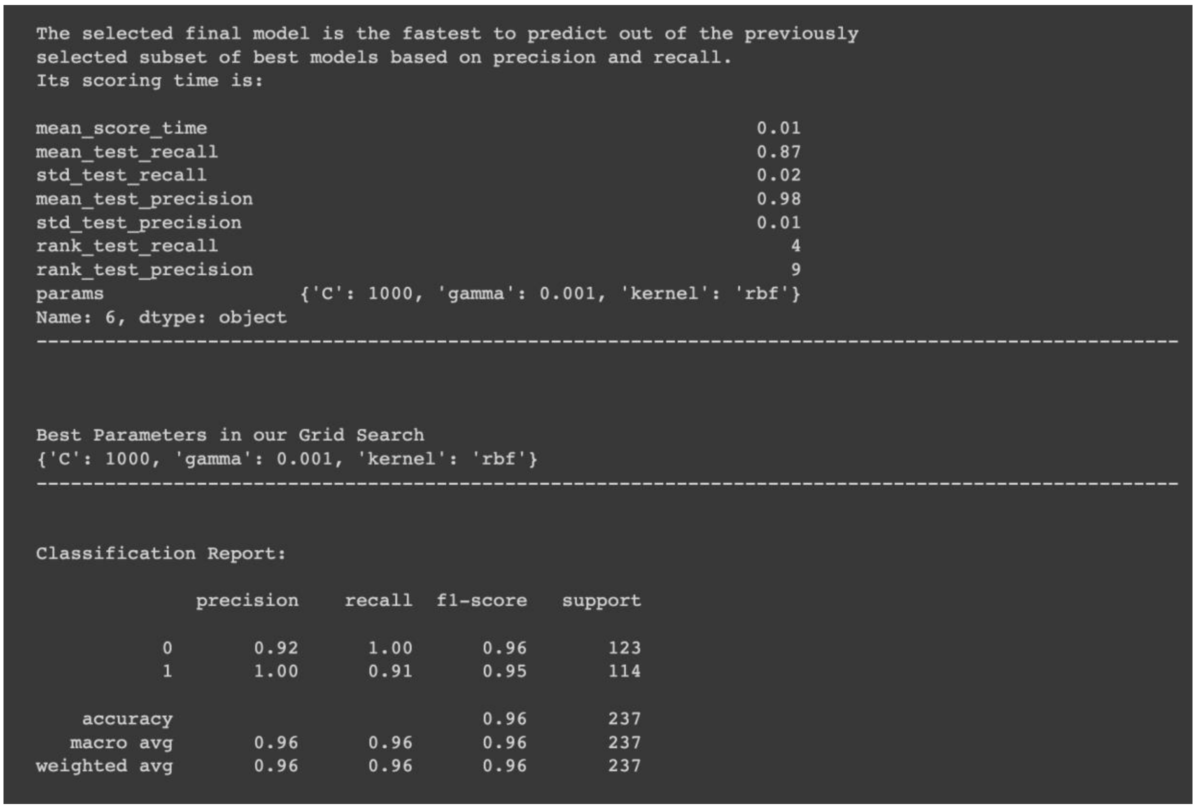
```
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.12.2)
Requirement already satisfied: numpy!=1.24.0,>=1.17 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.22.4)
Requirement already satisfied: pandas>=0.25 in /usr/local/lib/python3.10/dist-packages (from seaborn) (1.5.3)
Requirement already satisfied: matplotlib!=3.6.1,>=3.1 in /usr/local/lib/python3.10/dist-packages (from seaborn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib!=3.6.1,>=3.1->seaborn) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.25->seaborn) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib!=3.6.1,>=3.1->seaborn) (1.16.0)
```

Fig.2(a) OUTPUT OF THE PACKAGES IMPORT



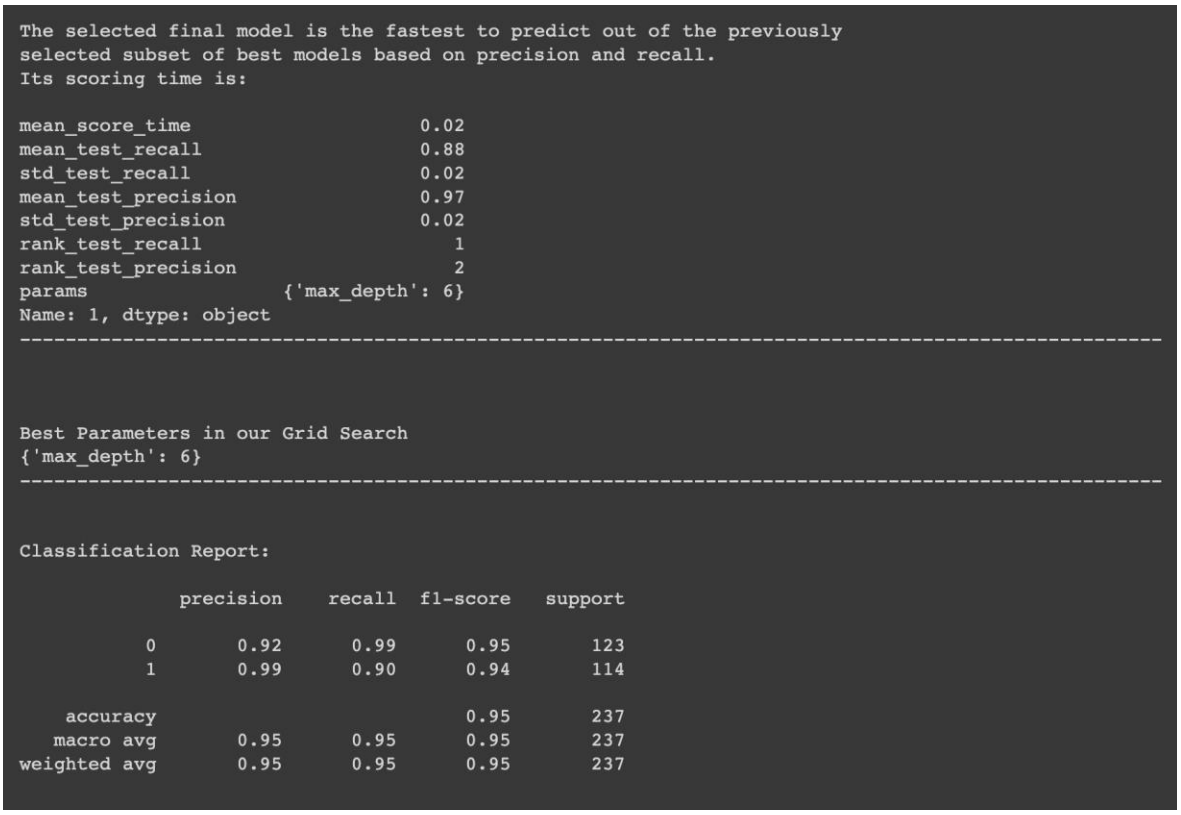
```
              precision    recall  f1-score   support

           0       1.00      0.65      0.79       732
           1       0.45      1.00      0.62       214

    accuracy                           0.73       946
   macro avg       0.73      0.82      0.70       946
weighted avg       0.88      0.73      0.75       946
```

Fig.2(b) OUTPUT OF DATA CLUSTERING



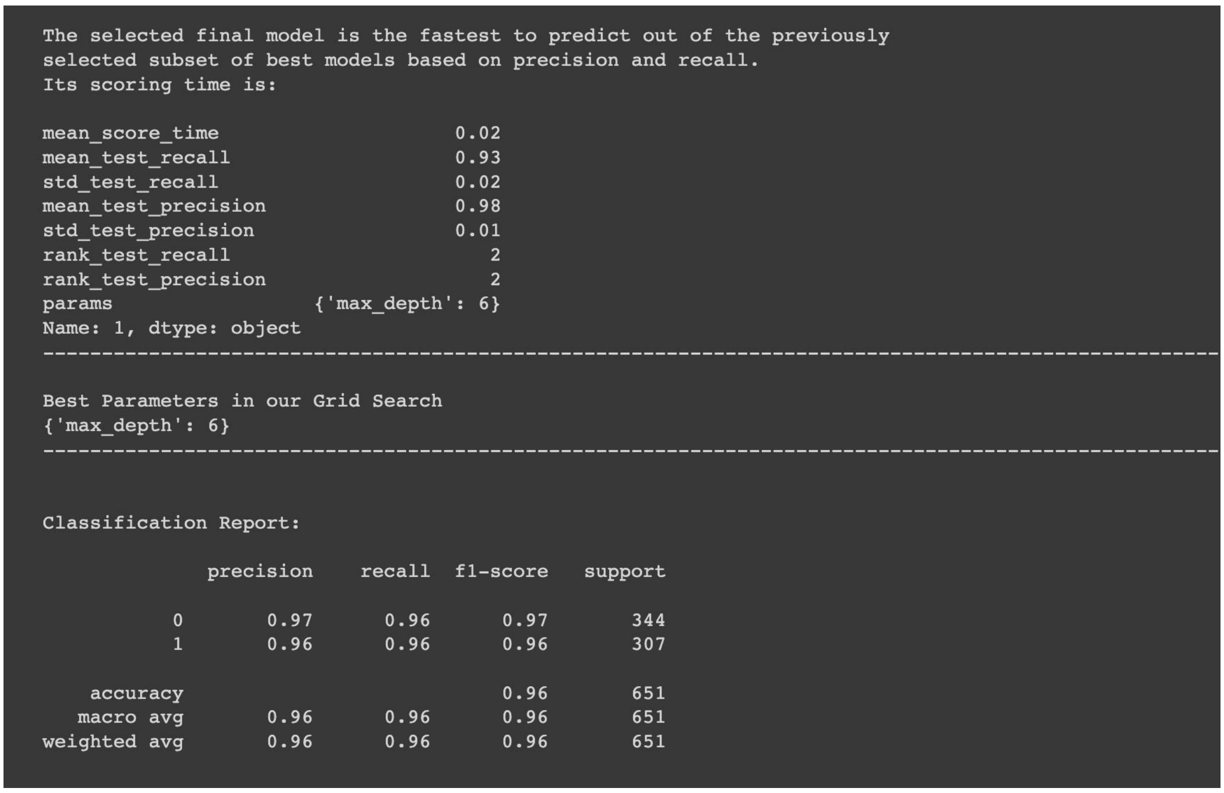
```
The selected final model is the fastest to predict out of the previously
selected subset of best models based on precision and recall.
Its scoring time is:

mean_score_time                                              0.01
mean_test_recall                                             0.87
std_test_recall                                              0.02
mean_test_precision                                          0.98
std_test_precision                                           0.01
rank_test_recall                                                4
rank_test_precision                                             9
params                {'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
Name: 6, dtype: object
----------------------------------------------------------------------------------------------------



Best Parameters in our Grid Search
{'C': 1000, 'gamma': 0.001, 'kernel': 'rbf'}
----------------------------------------------------------------------------------------------------


Classification Report:

              precision    recall  f1-score   support

           0       0.92      1.00      0.96       123
           1       1.00      0.91      0.95       114

    accuracy                           0.96       237
   macro avg       0.96      0.96      0.96       237
weighted avg       0.96      0.96      0.96       237
```

Fig.2(c) OUTPUT OF SVC MODEL



```
The selected final model is the fastest to predict out of the previously
selected subset of best models based on precision and recall.
Its scoring time is:

mean_score_time                   0.02
mean_test_recall                  0.88
std_test_recall                   0.02
mean_test_precision               0.97
std_test_precision                0.02
rank_test_recall                     1
rank_test_precision                  2
params                {'max_depth': 6}
Name: 1, dtype: object
----------------------------------------------------------------------------------------------------



Best Parameters in our Grid Search
{'max_depth': 6}
----------------------------------------------------------------------------------------------------


Classification Report:

              precision    recall  f1-score   support

           0       0.92      0.99      0.95       123
           1       0.99      0.90      0.94       114

    accuracy                           0.95       237
   macro avg       0.95      0.95      0.95       237
weighted avg       0.95      0.95      0.95       237
```

Fig.2(d) OUTPUT OF RANDOMFOREST MODEL



```
The selected final model is the fastest to predict out of the previously
selected subset of best models based on precision and recall.
Its scoring time is:

mean_score_time                   0.02
mean_test_recall                  0.93
std_test_recall                   0.02
mean_test_precision               0.98
std_test_precision                0.01
rank_test_recall                     2
rank_test_precision                  2
params                {'max_depth': 6}
Name: 1, dtype: object
----------------------------------------------------------------------------------------------------

Best Parameters in our Grid Search
{'max_depth': 6}
----------------------------------------------------------------------------------------------------


Classification Report:

              precision    recall  f1-score   support

           0       0.97      0.96      0.97       344
           1       0.96      0.96      0.96       307

    accuracy                           0.96       651
   macro avg       0.96      0.96      0.96       651
weighted avg       0.96      0.96      0.96       651
```

Fig.2(e) OUTPUT OF OVERALL ANALYSIS

Chapter-4

CONCLUSION AND FUTURE WORK

4.2 RESEARCH GAP:

There are several research gaps in the area of online fraud detection. Some ofthe most pressing gaps include

1) The need for more comprehensive data sets. Most existing fraud detection models are trained on data sets that are either too small or too narrow in scope. This makes it difficult for these models to generalize to new types of fraud or new environments.
2) The need for more robust and accurate models. Current fraud detection models are often not very accurate. This is due to several factors, including the complexity of fraud, the ever-changing nature of fraud, andthe limited availability of data.
3) The need for more efficient and scalable models. Fraud detection models need to be able to process large volumes of data in real time. This is a challenge for many models, as they can be computationally expensive and difficult to scale.
4) The need for more effective prevention strategies. Fraud detection is only one part of the fight against fraud. It is also important to develop effective prevention strategies that can stop fraud before it happens.

5) The increasing sophistication of fraudsters. Fraudsters are constantly developing new ways to commit fraud, which makes it difficult for fraud detection models to keep up.
6) The growing volume of data. The amount of data that is generated online is growing exponentially. This makes it difficult for fraud detection models to process all of the data promptly.
7) The need for collaboration. Fraud is a global problem that requires a global solution. This means that businesses, governments, and law enforcement agencies need to work together to share information and resources to combat fraud.

4.1 RESULT, CONCLUSION, AND INFERENCE:

Online fraud detection is a crucial aspect of e-commerce and financial services. By analyzing user behavior and transaction patterns, fraud detection systems can identify suspicious activity and prevent fraudulent transactions from occurring.

The result of implementing an online fraud detection system is a reduction in financial losses due to fraud. This not only saves money for businesses but also helps to maintain customer trust and loyalty.

Based on the data collected by the system, conclusions can be drawn about the types of fraudulent activities that are most prevalent. This information can be used to improve the fraud detection system and prevent future fraud attempts.

Overall, the inference is that online fraud detection is an essential tool for businesses operating in the digital space. By implementing an effective fraud detection system, businesses can protect themselves and their customers from financial losses and maintain a secure online environment.

REFERENCES

1. https://www.kaggle.com/
2. https://ieeexplore.ieee.org/document/78471363.
3. https://towardsdatascience.com/credit-card-fraud-detection-using-machine-learning-python-5b098d4a8edc?gi=cf11e90c4cec
4. https://www.sciencedirect.com/science/article/pii/S187705092030065X
5. https://www.infosysbpm.com/blogs/bpm-analytics/machine-learning-for-credit-card-fraud-detection
6. Anderson M. (2008). ‗From Subprime Mortgages to Subprime Credit Cards'. Communities and Banking, Federal Reserve Bank of Boston, pp. 21-23.
7. Anwer et al. (2009-2010). ‗Online Credit Card Fraud Prevention System for Developing Countries', International Journal of Reviews in Computing, ISSN: 2076-3328, Vol. 2, pp. 62-70.
8. Chang C. & Chang S. (2010). ‗The Design of E-Traveler's Check with efficiency and Mutual Authentication'. Journal of Networks, Vol. 5, No. 3, pp. 275-282
9. Dharwa J.N. & Patel A.R. (2011). ‗A Data mining with Hybrid Approach Based Transaction Risk Score Generation Model (TRSGM) for Fraud Detection of Online Financial Transaction'. International Journal of Computer Applications, Vol. 6, No. 1, pp. 18-25.
10. Kumar D. & Ryu Y. (2009). ‗A Brief Introduction of Biometrics and Fingerprint Payment Technology '. International Journal of Advanced Science and Technology', Vol. 4, pp. 25-38

DEMONSTRATION VIDEO:

https://drive.google.com/drive/folders/1gWci__oZvNnfGHP9v0yyh5QUblsbcYRX

BIO-DATA

	Name: Nikhil Chordia A Mobile Number: 7010593392 E-mail: nikhilchordia.a2021@vitstudent.ac.in Permanent Address: 112 kalavai street, Chintadripet Chennai-600002
	Name: Hemant M Mehta Mobile Number: 8610064857 E-mail: hemant.mmehta2021@vitstudent.ac.in Permanent Address: Arihant Orchid apt,54 flowers road, 9th floor 9C, kilpauk Chennai- 600010
	Name: Sujay Ghosh Mobile Number: 9434481092 E-mail: sujay.ghosh2021@vitstudent.ac.in Permanent Address: G-100, Sukanta Nagar, Salt Lake, Bidhannager, Chringrighata, Kolkata-700098