CAPSTONE PROJECT

COMPREHENSIVE ANALYSIS OF FIND-DEFAULT (PREDICTION OF CREDIT CARD FRAUD) CREDIT CARD FRAUD DETECTION FOR A HIGHLY IMBALANCED DATASET USING VARIOUS MACHINE LEARNING MODELS. SUCH AS

1. LINEAR REGRESSION
2. RANDOM FOREST CLASSIFIER
3. DECISION TREE
4. SUPPORT VECTOR MACHINE
5. GRADIENT BOOSTING

COMPREHENSIVE PROJECT REPORT SUBMITTED FOR THE PROGRAM COMPLETION BOOT CAMP MI AL CERTIFICATION PROGRAM

ON

CREDIT CARD FRAUD DECEPTION

SUBMITTED TO:

UPGRAD-KNOWLEDGE HUT

Declaration

I, Chhaya Nikam hereby declare that the work presented in this project submission, titled **"Capstone Project-Credit Card Fraud Detection"**, is entirely my own effort unless otherwise indicated in the references. All sources used for information, data, or code have been duly cited and acknowledged in the bibliography section of this submission.

I affirm that this project represents original work conducted by me during my tenure at Upgrad.

I further declare that this project submission has not been previously submitted for academic evaluation or assessment in any other institution or context. Additionally, I understand the academic integrity policies of Upgrad, and I have adhered to them throughout the completion of this project.

Date: [18-May-2024]

Signature: Chhaya

Project Summary:

Title: Evaluating Various Machine Learning Models for Credit Card Fraud Detection

Introduction

Credit card fraud is a significant concern for financial institutions and cardholders worldwide. Machine learning techniques offer effective solutions for detecting fraudulent transactions in real-time. In this project, we aim to develop and compare several machine learning models to identify fraudulent activities in credit card transactions.

Objective:

The primary objective of this project is to evaluate the performance of different machine learning algorithms in detecting credit card fraud. Specifically, we will implement the following models:

1. Linear Regression
2. Random Forest Classifier
3. Decision Tree
4. Support Vector Machine (SVM)
5. Gradient Boosting

Methodology:

1. **Data Preprocessing**: We will preprocess the credit card transaction data, including handling missing values, scaling numerical features, and encoding categorical variables if applicable.

2. **Model Implementation**: Each machine learning model will be implemented using appropriate libraries such as scikit-learn or Numpy, Pandas etc. We will tune hyperparameters using techniques like grid search or random search to optimize model performance.

3. **Training and Evaluation:** The models will be trained on a training dataset and evaluated using a separate validation set. We will assess their performance based on metrics such as accuracy, precision, recall, and F1-score. Additionally, we will consider the area under the ROC curve (AUC-ROC) to account for class imbalance.

4. **Comparison**: We will compare the performance of each model based on evaluation metrics. Insights gained from this comparison will guide the selection of the most effective algorithm for credit card fraud detection.

Results:

The results obtained from our experiments will provide valuable insights into the effectiveness of different machine learning models for credit card fraud detection. We will present comparative analyses of model performance and discuss the strengths and limitations of each approach.

Conclusion:

In conclusion, this project aims to contribute to the advancement of credit card fraud detection techniques by evaluating and comparing various machine learning models. The findings will inform financial institutions and stakeholders in selecting the most suitable algorithm for detecting fraudulent transactions, ultimately enhancing security and trust in the financial ecosystem.

The "best" machine learning model for credit card fraud detection can depend on various factors, including the specific characteristics of the dataset, the available computational resources, and the desired balance between interpretability and performance. However, some commonly used and effective models for this task include:

1. Random Forest

- **Pros**:
  - Handles imbalanced data well
  - Provides feature importance
  - Robust to overfitting with proper tuning
- **Cons**:
  - Can be computationally expensive

```
Accuracy Output
1781/1781 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step -
accuracy: 0.9993 - loss: 0.0032
```

- `Test Loss: 0.0029`

- `Test Accuracy: 0.9993`

2. Gradient Boosting Machines (GBMs)

- Examples: XGBoost, LightGBM, CatBoost
- **Pros**:
    - High predictive power
    - Effective handling of imbalanced datasets
    - Customizable loss functions
- **Cons**:
    - Can be slow to train
    - More complex to tune than simpler models
- `Accuracy: 0.9989466661985184`

3. Logistic Regression

- **Pros**:
    - Simple and interpretable
    - Fast to train and deploy
    - Good baseline performance
- **Cons**:
    - May not capture complex relationships in the data
- `Accuracy score on Test Data :  0.9289340101522843`

4. Neural Networks

- Examples: Deep learning models with architectures like MLP (Multilayer Perceptron)
- **Pros**:
    - Capable of capturing complex patterns
    - Scalable to large datasets
- **Cons**:

    - Requires more data and computational power
    - Harder to interpret

5. Support Vector Machines (SVM)

- **Pros**:
    - Effective in high-dimensional spaces
    - Can be robust with the right kernel
- **Cons**:
    - Computationally intensive
    - Less interpretable than tree-based models

Considerations for Model Selection

1. **Imbalanced Data Handling**: Credit card fraud datasets are typically highly imbalanced, with fraudulent transactions being a small minority. Techniques like oversampling, undersampling, and using specialized algorithms (e.g., SMOTE for oversampling) can help address this.
2. **Feature Engineering**: Domain-specific features, such as transaction frequency, amount, and user behavior patterns, can significantly enhance model performance.
3. **Evaluation Metrics**: Precision, recall, F1-score, and area under the precision-recall curve (AUC-PR) are more informative than accuracy in the context of fraud detection due to the class imbalance.
4. **Explainability**: In financial services, regulatory requirements might necessitate model interpretability, making simpler models like logistic regression or decision trees more appealing despite potentially lower performance.

Recommendation

For a balanced approach, starting with a **tree-based ensemble method like Random Forest or Gradient Boosting (**e.g., XGBoost) is often a good choice due to their robustness and strong performance. These models can be followed by more sophisticated methods like neural networks if computational resources allow and if the additional complexity is justified by improved performance. Always validate the model with appropriate cross-validation techniques and evaluate using metrics suited to imbalanced data.

References:

Dataset used: creditcard.csv

By conducting a comprehensive evaluation of multiple machine learning algorithms, this project seeks to contribute to the ongoing efforts to combat credit card fraud and safeguard the interests of consumers and businesses alike.

Data Exploration and Analysis:

• Exploratory Data Analysis (EDA) is a crucial step in any machine learning project as it helps in understanding the dataset, identifying patterns, and formulating hypotheses. In each of the model various exploratory data analysis techniques are performed on a given dataset..

1. **Summary Statistics:**

   - Calculate basic statistics such as mean, median, mode, standard deviation, range, and quartiles for numerical features.
   - Count the frequency of categories for categorical features.

2. **Data Visualization:**

   - Histograms: Visualize the distribution of numerical features.
   - Box plots: Identify outliers and understand the spread of data.
   - Bar plots: Show the frequency of different categories in categorical features.
   - Scatter plots: Examine the relationship between two numerical features.
   - Pair plots: Visualize pairwise relationships between multiple numerical features.
   - Heatmaps: Show the correlation between numerical features.
   - Time series plots: Analyze trends, seasonality, and anomalies in time-series data.

Data Preprocessing:

• Identify missing values in the dataset and decide on strategies for handling them (e.g., imputation, deletion, or interpolation) if applicable has been applied

- Identify outliers using visualization techniques or statistical methods such as z-score, IQR (Interquartile Range), or isolation forest.

Feature Engineering:

- Create new features based on existing ones (e.g., feature scaling, binning, encoding categorical variables, creating interaction terms).
- Analyze the distribution of engineered features and their relationship with the target variable.

Model Selection:

• Choose machine learning algorithms or statistical models based on the problem type and data characteristics. • Consider the trade-offs between interpretability and predictive power.

Model Training and Evaluation:

• Split the data into training and test sets, train the model on the training data, and evaluate its performance using relevant metrics.

Model Interpretability:

• Emphasize model interpretability, especially if the project has real-world implications.

• Use feature importance to explain model predictions.

Organizing Assets -steps below

Open Anaconda terminal

1. Created the project folder
   mkdir credit_card_fraud_detection
   cd credit_card_fraud_detection

2. Created the subfolders
   Mkdir data_model_visuals

3. Created the ReadMe.md file
   Touch ReadMe.md

Libraries used in various Models:

1. pandas

Pandas is an open-source data manipulation and analysis library for Python, providing powerful data structures like Series and DataFrame. It facilitates data cleaning, transformation, and analysis with an extensive set of functions for handling real-world data efficiently.

2. Numpy

NumPy is an open-source library for Python that supports large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

NumPy enables high-performance operations on large datasets and serves as the foundation for other scientific libraries like pandas and SciPy.

3. Matplotlib

Is an open-source plotting library for Python that enables the creation of static, interactive, and animated visualizations in various formats. It provides a wide range of plotting functions to generate high-quality graphs and charts, such as line plots, scatter plots, bar charts, and histograms.

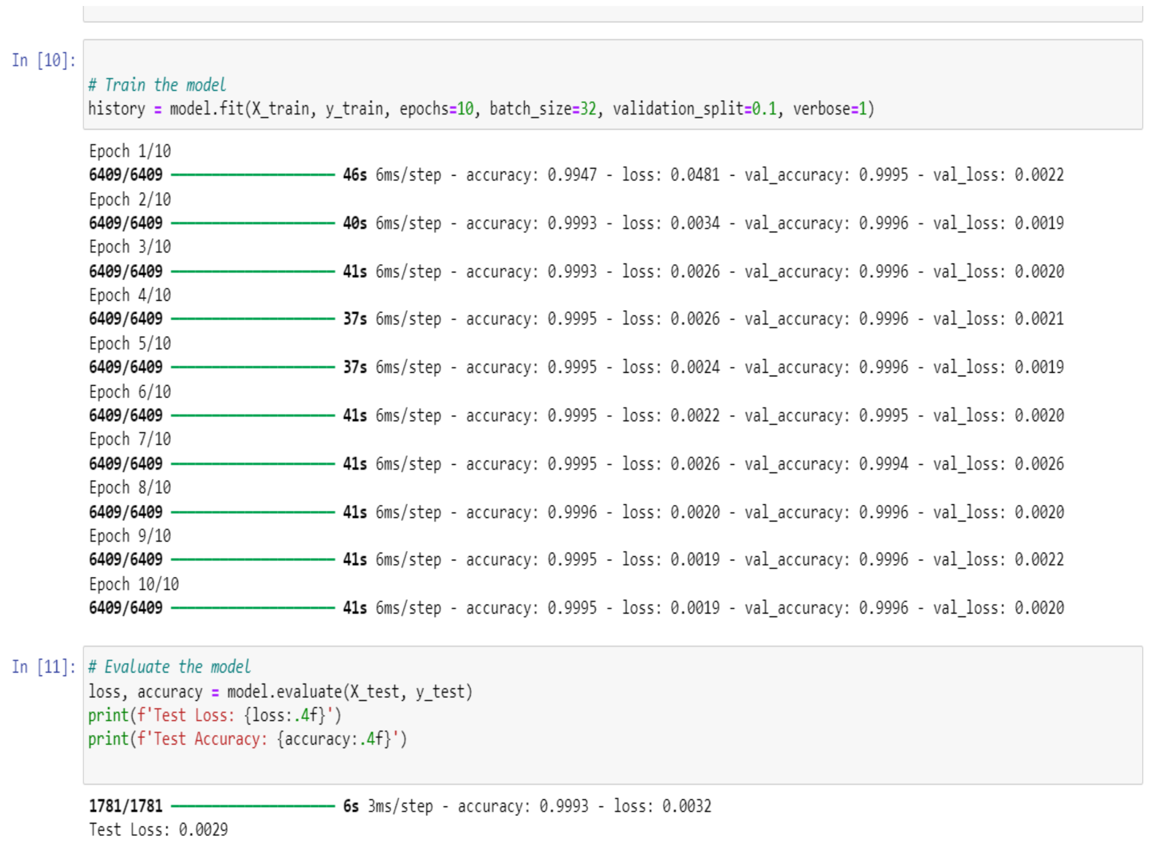
4. Seaborn

It is a Python data visualization library based on matplotlib that provides a high-level interface for drawing attractive and informative statistical graphics. It simplifies the creation of complex visualizations with minimal code, offering a variety of plots such as heatmaps, violin plots, and pair plots. Seaborn enhances matplotlib's capabilities with features like automatic estimation and plotting of linear regression models, making it a powerful tool for exploratory data analysis and presentation of statistical result

5.Scikit-learn (sklearn)

Is a Python library for machine learning built on NumPy, SciPy, and matplotlib. It provides simple and efficient tools for data mining and data analysis, including supervised and unsupervised learning algorithms. Scikit-learn is designed to interoperate with other Python libraries and tools, making it easy to integrate into data science workflows for tasks like classification, regression, clustering, and dimensionality reduction.

RNN -LSTM Accuracy output



```
In [10]:
# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.1, verbose=1)

Epoch 1/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 46s 6ms/step - accuracy: 0.9947 - loss: 0.0481 - val_accuracy: 0.9995 - val_loss: 0.0022
Epoch 2/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 40s 6ms/step - accuracy: 0.9993 - loss: 0.0034 - val_accuracy: 0.9996 - val_loss: 0.0019
Epoch 3/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9993 - loss: 0.0026 - val_accuracy: 0.9996 - val_loss: 0.0020
Epoch 4/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 37s 6ms/step - accuracy: 0.9995 - loss: 0.0026 - val_accuracy: 0.9996 - val_loss: 0.0021
Epoch 5/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 37s 6ms/step - accuracy: 0.9995 - loss: 0.0024 - val_accuracy: 0.9996 - val_loss: 0.0019
Epoch 6/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9995 - loss: 0.0022 - val_accuracy: 0.9995 - val_loss: 0.0020
Epoch 7/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9995 - loss: 0.0026 - val_accuracy: 0.9994 - val_loss: 0.0026
Epoch 8/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9996 - loss: 0.0020 - val_accuracy: 0.9996 - val_loss: 0.0020
Epoch 9/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9995 - loss: 0.0019 - val_accuracy: 0.9996 - val_loss: 0.0022
Epoch 10/10
6409/6409 ━━━━━━━━━━━━━━━━━━━━ 41s 6ms/step - accuracy: 0.9995 - loss: 0.0019 - val_accuracy: 0.9996 - val_loss: 0.0020

In [11]: # Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy:.4f}')

1781/1781 ━━━━━━━━━━━━━━━━━━━━ 6s 3ms/step - accuracy: 0.9993 - loss: 0.0032
Test Loss: 0.0029
```