Github link - https://github.com/sbose03/CS210final

Online payment fraud is a growing concern in the financial sector, with significant costs for both financial institutions and consumers. Fraudulent transactions erode consumer confidence, cause financial losses, and undermine the reputation of financial service providers. This project addresses the problem by developing a machine learning-based fraud detection system to identify and mitigate online payment fraud.

The project focuses on analyzing transaction data to identify fraudulent patterns while addressing challenges like data imbalance and high false-positive rates. Since most transactions are legitimate, the dataset I had was largely imbalanced. The dataset consists of 10,000 rows of transaction data and was downloaded from Kaggle.

This project is important because it integrates data science, finance, and cybersecurity, addressing a critical issue in the digital economy. Online payment fraud detection systems currently face challenges with false positives, which can inconvenience customers and lead to attrition. Moreover, detecting rare fraudulent transactions in large datasets is difficult due to the inherent data imbalance. By exploring advanced techniques such as SMOTE for oversampling and neural networks for deep learning, this project goes beyond traditional fraud detection methods like logistic regression and decision trees. By combining these techniques, careful evaluation ensures that the proposed model achieves a balance between accuracy and speed, making it suitable for real-time fraud detection.

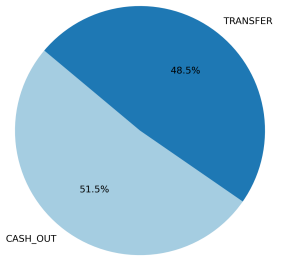
The dataset used in this project was sourced from Kaggle and comprises of 10,000 records of online payment transactions. The data includes features such as transaction types, amounts, balances, and labels indicating whether a transaction was fraudulent. Preprocessing steps included cleaning missing values, encoding categorical variables, and normalizing numerical features for consistency.

Exploratory data analysis was done to find patterns in transaction types and balances. For example, I discovered that fraudulent transactions only happened under ‘CASH_OUT’ and ‘TRANSFER’ transaction types.

The code that showed how fraudulent transactions only happened under ‘CASH_OUT’ and ‘TRANSFER’ transaction types.

```
import sqlite3
import pandas as pd
import matplotlib.pyplot as plt
# Connect to the SQLite database
conn = sqlite3.connect("onlinefraud.db")
# SQL query to count the different types of fraudulent transactions
query = """
SELECT type, COUNT(*) AS count
FROM fraud_transactions
WHERE isFraud = 1
GROUP BY type;
"""
# Execute the query and load the data into a pandas DataFrame
df = pd.read_sql_query(query, conn)
# Close the database connection
conn.close()
# Plot the pie chart
plt.figure(figsize=(6, 6))
plt.pie(df['count'], labels=df['type'], autopct='%1.1f%%', startangle=140, colors=plt.cm.Paired.colors)
plt.title("Distribution of Fraudulent Transaction Types")
plt.savefig("plot#2.png", dpi=300, bbox_inches="tight")
plt.show()
```

Plot:



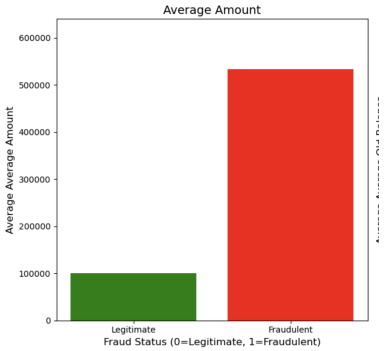
EDA also confirmed the imbalanced nature of the dataset, where fraudulent transactions constituted less than 1% of the total.

```
fraud_count = fraud['isFraud'].value_counts()
fraud_count
```

```
isFraud
0    9932
1      68
```

This necessitated the use of techniques like SMOTE for oversampling the minority class during model training. Additionally, I learned that fraudulent transactions had higher transaction amounts compared to legitimate transactions, and had lower new balance amounts compared to legitimate transactions.

Plot showing fraudulent transactions were for much larger amounts compared to legitimate transactions.



Several machine learning models were implemented and evaluated:

1. **Logistic Regression**: Used as a baseline model due to its simplicity and interpretability.
2. **Decision Trees**: Evaluated for their ability to capture non-linear patterns.
3. **Random Forests**: Tested with hyperparameter tuning and class weighting to handle data imbalance effectively.
4. **Neural Networks**: Used to explore the potential of deep learning for complex feature interactions.

Experimenting revealed that Random Forests, particularly with class weighting and SMOTE, outperformed the baseline model in terms of recall and AUC-ROC. Neural networks also showed promise, but ultimately fell short of the RandomForest+SMOTE model.

AUC-ROC score of RandomForest+SMOTE model

```
AUC-ROC Score: 0.9972278225806451
```

Code for the randomForest + SMOTE model:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report, roc_auc_score
from imblearn.over_sampling import SMOTE

# 1. Oversample the Minority Class (SMOTE)
smote = SMOTE(random_state=42)
X_train_resampled, y_train_resampled = smote.fit_resample(X_train, y_train)

# 2. Adjust Class Weights
rf_model = RandomForestClassifier(random_state=42, class_weight={0: 1, 1: 10})

# 3. Perform Hyperparameter Tuning with GridSearchCV
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5, 10]
}

grid_search = GridSearchCV(estimator=rf_model,
                           param_grid=param_grid,
                           scoring='roc_auc',
                           cv=5,
                           verbose=1)

grid_search.fit(X_train_resampled, y_train_resampled)

# Use the best model
best_rf_model = grid_search.best_estimator_

# 4. Evaluate the Best Model
rf_preds = best_rf_model.predict(X_test)
rf_probs = best_rf_model.predict_proba(X_test)[:, 1]

print("Random Forest Report:")
print(classification_report(y_test, rf_preds))
print(f"AUC-ROC Score: {roc_auc_score(y_test, rf_probs)}")

# 5. Adjust Prediction Threshold for Recall
threshold = 0.3
rf_preds_threshold = (rf_probs >= threshold).astype(int)
print("Random Forest Report (Adjusted Threshold):")
print(classification_report(y_test, rf_preds_threshold))
print(f"AUC-ROC Score (Adjusted Threshold): {roc_auc_score(y_test, rf_probs)}")
```

The project confirmed the hypothesis that advanced machine learning techniques can effectively detect fraudulent transactions. Random Forests achieved a high AUC-ROC score of 0.997, demonstrating its skill in handling imbalanced data. Adjusting the prediction threshold further enhanced recall for fraud cases, reducing the likelihood of missed fraudulent transactions.

Neural networks, while less interpretable than Random Forests, achieved slightly worse, showcasing its limitations. However, with more data, neural networks could be comparable or even better than the randomForest+SMOTE model.

Accuracy of the neural network model

```
Accuracy: 0.992
```

Advantages:

- High recall and precision rates ensured effective fraud detection with minimal false positives.
- Robust handling of imbalanced data through SMOTE and randomForest
- Scalability for large datasets, making the system suitable for real-time applications.

Limitations:

- Neural Networks can be computationally demanding
- Lack of location data could restrict the model's adaptability.

- The reliance on synthetic oversampling might introduce biases in the training process.

The project changed from the original proposal. Initially, the focus was limited to simpler models like logistic regression and decision trees. However, experimentation revealed the need for more advanced techniques like Random Forests and neural networks to achieve better performance. Handling data imbalance was more challenging than I thought, which is why I integrated SMOTE.

Bottlenecks included data and computing limitations for training neural networks and tuning hyperparameters for Random Forests. This project demonstrated the effectiveness of machine learning in detecting fraudulent online payment transactions. By leveraging a combination of models, data preprocessing techniques, and evaluation metrics, the system achieved high performance while addressing challenges like data imbalance and false positives.

Future work could explore the integration of domain-specific features and the use of the model in real-world environments to validate its real-world applicability.