FindDefault: Credit Card Fraud Prediction

Introduction

Credit cards are essential tools for online transactions, offering convenience for managing personal finances. However, this convenience is countered by the risk of credit card fraud, which can lead to significant financial losses for both customers and financial institutions. The **FindDefault** project seeks to mitigate these risks by developing a predictive model to identify fraudulent credit card transactions. Our objective is to create a robust classification model that accurately distinguishes between legitimate and fraudulent transactions, thereby aiding credit card companies in reducing losses and enhancing security.

Dataset Description

The dataset used for this project comprises credit card transactions made by European cardholders over a two-day period in September 2013. It contains 284,807 transactions, of which 492 are fraudulent. This results in a highly imbalanced dataset, with fraudulent transactions representing only 0.172% of the total.

Key Characteristics:

- **Total Transactions:** 284,807
- **Fraudulent Transactions:** 492
- **Class Imbalance:** Fraudulent transactions make up 0.172% of the dataset.

The dataset includes features such as transaction amount, timestamp, and anonymized numerical variables to protect sensitive information.

Project Pipeline

The project is divided into several key phases:

1. Exploratory Data Analysis (EDA)

- **Objective:** To understand the dataset and uncover patterns, relationships, and trends.
- **Steps:**
  - Load and preview the dataset.
  - Generate summary statistics to understand central tendencies and distributions.
  - Visualize data using histograms, box plots, scatter plots, and heatmaps to explore relationships and anomalies.
  - Identify patterns or correlations related to the fraudulent transactions.

2. Data Cleaning and Preprocessing

- **Objective:** Prepare the dataset for modeling by addressing data quality issues.
- **Steps:**
  - Handle missing values through imputation or removal.
  - Identify and address outliers to prevent them from skewing results.
  - Standardize or normalize numerical features to ensure uniformity.
  - Encode categorical variables using techniques like one-hot encoding or label encoding.

3. Handling Imbalanced Data

- **Objective:** To address the imbalance between fraudulent and non-fraudulent transactions.
- **Approaches:**
  - **Undersampling:** Reduce the number of non-fraudulent transactions.
  - **Oversampling:** Increase the number of fraudulent transactions through methods like Random Oversampling.
  - **SMOTE (Synthetic Minority Over-sampling Technique):** Generate synthetic samples of the minority class to balance the dataset.
  - **ADASYN (Adaptive Synthetic Sampling):** Create synthetic samples focusing on regions with a low density of minority class samples.

4. Feature Engineering

- **Objective:** Improve model performance by creating and transforming features.
- **Steps:**
  - Develop new features that highlight significant patterns or relationships.
  - Apply dimensionality reduction techniques such as PCA if needed.
  - Transform features to handle skewness, normalize distributions, or encode categorical data.

5. Model Selection

- **Objective:** Identify and select appropriate machine learning models for the classification task.
- **Models Evaluated:**

1. Logistic Regression

Explanation: Logistic Regression is a statistical model used for binary classification tasks. It predicts the probability that a given input belongs to a specific class (fraudulent or legitimate) by applying a logistic function to a linear combination of input features. The output is a probability score between 0 and 1, which is then thresholded to make a binary decision.

Usage: In FindDefault project, Logistic Regression was selected due to its simplicity and interpretability. It provided clear insights into how features influence the prediction of fraudulent transactions. The model was trained using SMOTE to handle class imbalance, resulting in high performance with an ROC-AUC score of 0.99 on the training set and 0.97 on the test set. Its lower computational requirements made it suitable for real-time deployment.

Benefits:

Interpretability: The model is highly interpretable, providing insights into the impact of each feature on the likelihood of fraud.

Efficiency: Logistic Regression is computationally efficient and works well with large datasets, making it a good starting point for baseline comparisons.

Probability Estimates: It outputs probabilities, enabling threshold adjustments for different risk levels.

2. XGBoost

Explanation: XGBoost (Extreme Gradient Boosting) is an ensemble learning method that builds multiple decision trees sequentially. Each tree tries to correct the errors of the previous ones, resulting in a strong predictive model. XGBoost is known for its speed, accuracy, and ability to handle large datasets and high-dimensional data.

Usage: XGBoost was considered in the FindDefault project as one of the potential models due to its ability to capture complex relationships in the data. It performed well during evaluation but was not chosen as the final model due to higher computational costs and the added complexity in interpretation and deployment. However, it remains a powerful option for scenarios where slight improvements in prediction accuracy are crucial, and resources are available to support its deployment.

Benefits:
Intuitive Visualization: The tree structure is easy to visualize and explain to non-technical stakeholders.
Feature Importance: It provides a clear ranking of feature importance, helping in feature selection and understanding which factors contribute most to fraud detection.
Flexibility: The model can handle both numerical and categorical data, making it versatile.

3. Decision Tree

Explanation: A Decision Tree is a non-parametric model that splits the dataset into subsets based on feature values, forming a tree-like structure. Each node represents a feature, and each branch represents a decision rule, ultimately leading to a classification at the leaf nodes. Decision Trees are easy to understand and interpret but may suffer from overfitting if not properly pruned.

Usage: In the FindDefault project, Decision Trees were evaluated as a model choice due to their ability to model complex patterns in the data. They provided valuable insights during the exploratory phase but were ultimately not selected as the final model. While Decision Trees offer interpretability, they can be prone to overfitting, especially in high-dimensional datasets. Nevertheless, they served as a useful comparison against more complex models like XGBoost and simpler ones like Logistic Regression.

Benefits:

- **Performance:** XGBoost is known for its high predictive performance and is often the top choice in machine learning competitions.
- **Handling Missing Data:** It automatically handles missing values, simplifying the preprocessing stage.
- **Customizability:** The model provides extensive hyperparameter tuning options, allowing fine-grained control over its behavior.

6. Model Training and Validation

- **Objective:** Train the model on the training dataset and validate its performance.
- **Steps:**
  - Split the dataset into training and testing sets.
  - Train models and perform cross-validation to ensure robustness.
  - Use techniques like GridSearchCV to optimize hyperparameters.

Hyperparameter Tuning

To enhance model performance, hyperparameters were meticulously tuned using GridSearchCV. This technique involves systematically exploring a specified range of hyperparameters for each model to identify the optimal combination that maximizes performance metrics. GridSearchCV evaluates multiple parameter configurations by performing cross-validation, ensuring the selected hyperparameters provide the best trade-off between model accuracy and generalization.

Choosing the Best Model

After tuning hyperparameters, the performance of various models was assessed, and the **Logistic Regression** model emerged as the top performer. This model, trained on a dataset balanced using SMOTE, demonstrated outstanding performance with an ROC-AUC score of 0.99 on the training set and 0.97 on the test set. These high scores indicate the model's excellent ability to differentiate between fraudulent and legitimate transactions.

The decision to select Logistic Regression was based on several factors:

- **Simplicity:** The model's straightforward nature makes it easy to interpret and understand.
- **Ease of Interpretation:** Logistic Regression provides clear insights into the influence of each feature on the prediction, aiding in interpretability.
- **Lower Computational Resource Requirements:** Compared to more complex models like XGBoost, Logistic Regression is computationally efficient, making it suitable for deployment in real-time systems.

By combining hyperparameter tuning with the selected optimal model, the project ensures that the final Logistic Regression model is both highly accurate and practical for real-world application.

7. Model Evaluation

- **Objective:** Assess the model's performance using various metrics.
- **Metrics:**
  - **Accuracy:** Proportion of correctly classified transactions.
  - **Precision:** Proportion of true positives among all positive predictions.
  - **Recall:** Proportion of actual fraudulent transactions correctly identified.
  - **F1-Score:** Harmonic mean of precision and recall.
  - **ROC-AUC:** Area under the receiver operating characteristic curve.
- **Visualization:** Use confusion matrix to understand the model's error types.

8. Model Deployment

- **Objective:** Deploy the best model for real-time fraud detection.

- **Steps:**
    - **Serialize the model using pickle for future use.**

```python
import pickle

# Save the best model to a pickle file
with open('load_best_model.pkl', 'wb') as f:
  pickle.dump(logistic_bal_smote_model, f)
```

Load and use the model for predictions:

```python
# Load the pickled model
with open('load_best_model.pkl', 'rb') as f:
  model = pickle.load(f)

# Make predictions on new data
new_data = X_test
predictions = model.predict(new_data)
```

-
    - **Deploy the model to a production environment for real-time predictions.**

Streamlit Integration

Streamlit is used to create an interactive web application for real-time fraud detection. The app allows users to input transaction data and receive predictions.

Setting Up the Streamlit App

1. **Create a streamlit_app.py file** for the application.
2. **Implement the functionality** to input transaction data and display predictions.

Streamlit Application Code:

import streamlit as st

import pandas as pd

import pickle

from sklearn.preprocessing import StandardScaler, PowerTransformer

 # Function to load the model from a specified file path

def load_model(file_path):

```
    with open(file_path, 'rb') as file:
        model = pickle.load(file)
    return model

# Path to the model file
model_path = 'load_best_model.pkl'  # replace with the correct path to your best_model.pkl

# Load the model
model = load_model(model_path)
st.success("Model loaded successfully")

# Title of the app
st.write("""
# Credit Card Fraud Detection
This app predicts **Credit Card Fraud** based on transaction data!
""")

# Sidebar for user input features
st.sidebar.header('User Input Parameters')

def user_input_features():
    V1 = st.sidebar.slider('V1', -56.407510, 2.454930, 1.759061e-12)
    V2 = st.sidebar.slider('V2', -72.715728, 22.057729, -8.251130e-13)
    V3 = st.sidebar.slider('V3', -48.325589, 9.382558, -9.654937e-13)
    V4 = st.sidebar.slider('V4', -5.683171, 16.875344, 8.321385e-13)
    V5 = st.sidebar.slider('V5', -113.743307, 34.801666, 1.649999e-13)
    V6 = st.sidebar.slider('V6', -26.160506, 73.301626, 4.248366e-13)
    V7 = st.sidebar.slider('V7', -43.557242, 120.589494, -3.054600e-13)
    V8 = st.sidebar.slider('V8', -73.216718, 20.007208, 8.777971e-14)
    V9 = st.sidebar.slider('V9', -13.434066, 15.594995, -1.179749e-12)
```

```
V10 = st.sidebar.slider('V10', -24.588262, 23.745136, 7.092545e-13)
V11 = st.sidebar.slider('V11', -4.797473, 12.018913, 1.874948e-12)
V12 = st.sidebar.slider('V12', -18.683715, 7.848392, 1.053347e-12)
V13 = st.sidebar.slider('V13', -5.791881, 7.126883, 7.127611e-13)
V14 = st.sidebar.slider('V14', -19.214325, 10.526766, -1.474791e-13)
V15 = st.sidebar.slider('V15', -4.498945, 8.877742, -5.231558e-13)
V16 = st.sidebar.slider('V16', -14.129855, 17.315112, -2.282250e-13)
V17 = st.sidebar.slider('V17', -25.162799, 9.253526, -6.425436e-13)
V18 = st.sidebar.slider('V18', -9.498746, 5.041069, 4.950748e-13)
V19 = st.sidebar.slider('V19', -7.213527, 5.591971, 7.057397e-13)
V20 = st.sidebar.slider('V20', -54.497720, 39.420904, 1.766111e-12)
V21 = st.sidebar.slider('V21', -34.830382, 27.202839, -3.405756e-13)
V22 = st.sidebar.slider('V22', -10.933144, 10.503090, -5.723197e-13)
V23 = st.sidebar.slider('V23', -44.807735, 22.528412, -9.725856e-13)
V24 = st.sidebar.slider('V24', -2.836627, 4.584549, 1.464150e-12)
V25 = st.sidebar.slider('V25', -10.295397, 7.519589, -6.987102e-13)
V26 = st.sidebar.slider('V26', -2.604551, 3.517346, -5.617874e-13)
V27 = st.sidebar.slider('V27', -22.565679, 31.612198, 3.332082e-12)
V28 = st.sidebar.slider('V28', -15.430084, 33.847808, -3.518874e-12)
Amount = st.sidebar.slider('Amount', 0.000000, 25691.160000, 88.34962)

data = {
  'V1': V1, 'V2': V2, 'V3': V3, 'V4': V4, 'V5': V5, 'V6': V6, 'V7': V7,
  'V8': V8, 'V9': V9, 'V10': V10, 'V11': V11, 'V12': V12, 'V13': V13,
  'V14': V14, 'V15': V15, 'V16': V16, 'V17': V17, 'V18': V18, 'V19': V19,
  'V20': V20, 'V21': V21, 'V22': V22, 'V23': V23, 'V24': V24, 'V25': V25,
  'V26': V26, 'V27': V27, 'V28': V28, 'Amount': Amount
}
features = pd.DataFrame(data, index=[0])

return features
```

```
# Preprocessing function
def preprocess_data(df):
    # Assuming the scaler and power transformer were fit on the original training data
    scaler = StandardScaler()
    power_transformer = PowerTransformer(method='yeo-johnson', standardize=True)
     try:
        # Apply power transformation and scaling only to the Amount column
        df['Amount'] = power_transformer.fit_transform(df[['Amount']])
        df['Amount'] = scaler.fit_transform(df[['Amount']])
    except ValueError as e:
        st.error(f"Error in preprocessing data: {e}")
        return df

    return df

# Get user input features
df = user_input_features()
st.subheader('User Input Parameters')
st.write(df)

# Preprocess the user input features
preprocessed_data = preprocess_data(df)

# Make predictions
if model:
    prediction = model.predict(preprocessed_data)
    prediction_proba = model.predict_proba(preprocessed_data)

    st.subheader('Prediction')
    if prediction == 1:
```

```
        st.write('Fraudulent transaction')
    else:
        st.write('Not a fraudulent transaction')


    st.subheader('Prediction Probability')
    st.write(prediction_proba)
else:
    st.error("Model not loaded.")
```

Running the Streamlit App

- Install Streamlit if not already installed:

```
pip install streamlit
```

- Run the Streamlit app:

```
streamlit run streamlit_app.py
```

Evaluating and Selecting the Best Model for Balanced Data

We balanced the data using various techniques, including Undersampling, Oversampling, SMOTE, and ADASYN, and built several models such as Logistic Regression, XGBoost, Decision Tree.

All models showed good performance to varying extents. However, since Undersampling resulted in some loss of information, it is preferable to exclude these models from consideration.

The models trained with SMOTE and ADASYN performed well, but among them, the Logistic Regression model stood out. It achieved an impressive ROC score of 0.99 on the training set and 0.97 on the test set. This high ROC score indicates that Logistic Regression, enhanced with SMOTE, effectively distinguishes between classes.

Given its simplicity, ease of interpretation, and lower resource requirements compared to more complex models like Random Forest and XGBoost, the Logistic Regression model with SMOTE is deemed the best choice. Its efficiency and performance make it suitable for practical applications where both accuracy and resource constraints are critical.

Comprehensive Cost-Benefit Analysis

While most models performed well in terms of ROC score, Precision, and Recall, selecting the best model requires careful consideration of several factors, including infrastructure, resources, and computational power. Complex models like Random Forest, SVM, and XGBoost demand significant computational resources, leading to increased deployment costs. These models also come with disadvantages such as difficulty in interpretation and tuning, which adds to the complexity of model management and maintenance.

In contrast, simpler models like Logistic Regression are more cost-effective to build and deploy due to their lower computational requirements. Logistic Regression offers easier interpretation and implementation, which is crucial for understanding model decisions and ensuring regulatory compliance.

When evaluating the trade-offs between model complexity and performance, the financial implications of minor changes in the ROC score must be considered. If the monetary impact of a slight improvement in the ROC score is substantial, investing in a complex model might be justified despite higher costs. However, if the gains are marginal, a simpler, more cost-effective model like Logistic Regression is preferable due to its lower resource requirements and ease of use.

Conclusion

The **FindDefault** project developed a credit card fraud detection model by systematically addressing the challenges of identifying fraudulent transactions in a highly imbalanced dataset. The project followed a structured approach:

1. **Exploratory Data Analysis (EDA):** Provided insights into data distribution and relationships.
2. **Data Cleaning and Preprocessing:** Ensured data quality and readiness for modeling.
3. **Handling Imbalanced Data:** Used techniques like SMOTE and ADASYN to balance the dataset.

4. **Feature Engineering:** Enhanced model performance through feature creation and transformation.
5. **Model Selection and Training:** Evaluated various models and optimized hyperparameters.
6. **Model Evaluation:** Assessed model performance using multiple metrics.
7. **Model Deployment:** Serialized and deployed the model for real-time fraud detection.

Future Works

1. Integration of Real-Time Data Streams

- **Description:** Implementing a real-time data ingestion and processing pipeline to handle streaming credit card transaction data. This could involve integrating with tools like Apache Kafka or Spark Streaming to process transactions in real-time, allowing for immediate fraud detection.
- **Benefit:** Enhances the system's capability to detect and respond to fraudulent transactions as they happen, reducing financial loss and increasing the system's practical value.

2. Exploration of Deep Learning Models

- **Description:** Investigating the application of deep learning techniques such as Recurrent Neural Networks (RNNs) or Long Short-Term Memory (LSTM) networks, which could better capture temporal patterns and dependencies in transaction data.
- **Benefit:** These models might uncover complex patterns in the data that traditional machine learning models cannot, potentially improving fraud detection accuracy.

3. Incorporating Additional Data Sources

- **Description:** Expanding the dataset by incorporating external data sources, such as customer demographics, spending behavior patterns, or transaction metadata (e.g., device type, location data).
- **Benefit:** Additional contextual information could improve the model's ability to differentiate between fraudulent and legitimate transactions, especially in edge cases.

4. Model Explainability and Transparency

- **Description:** Implementing techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to make the model's decisions more interpretable to stakeholders.
- **Benefit:** Enhances trust and adoption of the model by providing insights into how and why certain predictions are made, which is particularly important in regulated industries.

5. Deployment of an AutoML Framework

- **Description:** Exploring the use of Automated Machine Learning (AutoML) frameworks to automate model selection, hyperparameter tuning, and feature engineering. This could include tools like Google AutoML, H2O.ai, or Azure AutoML.
- **Benefit:** Reduces the time and effort required for model development, while potentially uncovering better-performing models through automated processes.

6. Implementation of Anomaly Detection Techniques

- **Description:** Integrating unsupervised learning methods, such as Isolation Forests or Autoencoders, to detect anomalous transactions that may not have been labeled in the training data.
- **Benefit:** Improves the detection of novel or rare types of fraud that the supervised models may not have been trained to recognize.

7. Periodic Model Retraining and Monitoring

- **Description:** Setting up a system for continuous model monitoring and periodic retraining to adapt to changes in transaction patterns or emerging fraud tactics.
- **Benefit:** Ensures that the model remains accurate and effective over time, especially as fraudsters evolve their strategies.

Documentation Benefits to the Company

This comprehensive documentation offers several advantages:

1. **Improved Collaboration:** Ensures team members understand project objectives and methodologies.
2. **Streamlined Onboarding:** Facilitates quick integration of new team members.
3. **Enhanced Decision-Making:** Provides transparency for stakeholders to make informed decisions.
4. **Regulatory Compliance:** Demonstrates adherence to industry standards.

5. **Future-Proofing:** Lays a foundation for future work and integration of new technologies.