**Credit Card Fraud Detection**

**Brief Description**

This script is designed to detect fraudulent credit card transactions using various machine learning algorithms. The process involves multiple stages such as data loading, exploration, preprocessing, model training, hyperparameter tuning, evaluation, and model selection.

**1. Data Loading and Exploration**

* **Data Loading**: The dataset creditcard.csv is loaded into a pandas DataFrame. This dataset contains features such as transaction time, amount, and a binary target variable (Class), where 1 indicates a fraudulent transaction, and 0 indicates a legitimate one.
* **Exploration**: The basic properties of the dataset (number of rows, columns) are checked, and summary statistics are displayed. Visualizations are created to explore the distribution of features and detect any class imbalances (fraudulent vs. legitimate transactions).
* **Data Cleaning**: The script checks for missing values and duplicates to ensure the dataset is clean for model training.

**2. Feature Selection and Preprocessing**

* **Feature Selection**: Only relevant features like Time, Amount, and Class are retained for analysis.
* **Scaling**: The feature variables (X) are scaled using **StandardScaler** to standardize the data, making it suitable for training many machine learning models.
* **Train-Test Split**: The data is split into training and testing sets (80% for training and 20% for testing) using **scikit-learn's** train\_test\_split method.
* **Handling Class Imbalance**: Since fraud detection datasets are typically imbalanced, techniques like **undersampling** (via RandomUnderSampler) are applied to balance the class distribution.

**3. Hyperparameter Tuning with GridSearchCV**

The script performs **Grid Search Cross-Validation (GridSearchCV)** to optimize the hyperparameters of four machine learning models: Logistic Regression, Decision Tree, Random Forest, and Support Vector Classifier (SVC).

* **Logistic Regression**: Optimizes the regularization strength (C) and solver type (liblinear, saga).
* **Decision Tree**: Optimizes the tree depth (max\_depth), minimum samples required for a split (min\_samples\_split), and minimum samples required at a leaf node (min\_samples\_leaf).
* **Random Forest**: Optimizes the number of trees (n\_estimators), maximum tree depth (max\_depth), and minimum samples for splits and leaves.
* **SVC**: Optimizes the regularization parameter (C), kernel type (linear, rbf, poly), and the gamma parameter.

The best parameters for each model are found using **cross-validation** (with cv=5), ensuring the models generalize well to unseen data.

**4. Model Training and Evaluation**

Once the best hyperparameters are found, the models are trained on the training data. The evaluation involves various performance metrics:

* **Accuracy**: The overall proportion of correct predictions.
* **Precision and Recall**: Precision measures the percentage of true positive predictions out of all positive predictions, while recall measures the percentage of true positives out of all actual positives.
* **F1-Score**: The harmonic mean of precision and recall, which balances the two.
* **Confusion Matrix**: A matrix showing the true positives, true negatives, false positives, and false negatives.
* **ROC-AUC Score**: The area under the Receiver Operating Characteristic (ROC) curve, indicating how well the model discriminates between classes (fraud vs. non-fraud).

**5. Visualization of Results**

* **ROC Curve**: For each model, the ROC curve is plotted to visualize the trade-off between true positive rate (recall) and false positive rate.
* **Confusion Matrices**: Both the training and testing confusion matrices are visualized as heatmaps to see the distribution of true vs. predicted labels.
* **Bar Plot**: A bar plot shows the accuracy of each model, helping to compare their performance.

**6. Best Model Selection**

* **Model Comparison**: After evaluating all models, the one with the highest precision score is selected as the best model. The Random Forest classifier emerges as the best model based on its performance metrics.
* **Final Model Evaluation**: The best model is further evaluated on accuracy, precision, and recall. Additional metrics like confusion matrices and ROC curves are used to assess its performance.

**7. Model Saving**

* The **Random Forest model**, which performs best, is saved to disk using joblib for future use.

**8. Conclusions**

* **Performance**: The Random Forest Classifier, tuned with Grid Search, achieves an accuracy of around 93.8%, with a precision score of 97.5% and a perfect F1-score, demonstrating its effectiveness at detecting fraudulent transactions.
* **Insights from Data**:
  + Fraudulent transactions tend to involve smaller amounts (less than $2500).
  + There is no strong relationship between the time of day and the occurrence of fraud.
  + Fraud transactions are rare and need to be handled with techniques like **oversampling** or **undersampling** to avoid model bias towards the majority class.

**9. Brief Summary:**

 **Before Resampling:**

* **Logistic Regression**: High accuracy (99.92%) but poor performance on the minority class with low recall (0.61).
* **Decision Tree**: Similar high accuracy (99.92%) but lower performance on the minority class with recall around 0.75.
* **Random Forest**: Slightly better than Logistic Regression and Decision Tree, with a ROC-AUC score of 0.88 for imbalanced data.
* **Support Vector Classifier (SVC)**: Very poor performance with a ROC-AUC score of 0.5, indicating it failed to predict the minority class.

 **After Resampling** (Oversampling the minority class and undersampling the majority class):

* **Logistic Regression**: Accuracy dropped to 0.94, but performance improved for the minority class, with a recall of 0.92 and precision of 0.96.
* **Decision Tree**: Accuracy of 0.90, with balanced precision and recall.
* **Random Forest**: Similar improvement, with accuracy of 0.94 and high precision (0.99) for the minority class.
* **Support Vector Classifier (SVC)**: Still underperforming with an accuracy of 0.59 and poor recall.

 **After Hyperparameter Tuning**:

* **Logistic Regression**: The best parameters were C=0.1 and solver=liblinear. The model performed well with a precision of 0.96 and recall of 0.92.
* **Decision Tree**: Best parameters were max\_depth=5, min\_samples\_leaf=2, and min\_samples\_split=2. The model showed balanced performance with an accuracy of 0.91.
* **Random Forest**: Best model overall with accuracy of 0.94, high precision (0.99), and good recall for the minority class.

 **Best Model**: **Random Forest** was selected as the best model due to its high performance on both training and testing datasets. It showed perfect training performance and good test accuracy (0.94) with high precision and recall, indicating strong generalization capability.

**Summary of Results:**

* The **Random Forest Classifier** tuned with **Grid Search** is selected as the best-performing model based on test accuracy, precision, and recall.
* The model successfully detects fraud by identifying patterns in the transaction data, especially focusing on smaller transaction amounts.

**Key Takeaways:**

* **Handling class imbalance** is crucial in fraud detection to ensure models are not biased towards the majority class.
* **Hyperparameter tuning** improves the performance of machine learning models, making them more effective.
* **Random Forest** provides a robust solution, achieving high performance in fraud detection.

This code outlines an end-to-end process for **fraud detection in credit card transactions**, including **data exploration**, **preprocessing**, **model selection**, and **evaluation**, with **best model selection** and **hyperparameter tuning** for improved performance.