# Credit card Fraud Detection

1. **Data processing and data cleaning:**

**Data Size**: The dataset contains over 284,000 transactions, which makes it a moderately sized dataset. However, the dataset has 31 columns, which could result in a large number of features when using machine learning models. As such, you may need to consider feature selection techniques to reduce the dimensionality of the data.

**Class Imbalance**: The dataset is highly imbalanced, with only 0.172% of the transactions being fraudulent. This means that machine learning models trained on this dataset may have a high accuracy in detecting non-fraudulent transactions, but may perform poorly on fraudulent transactions. You may need to use techniques such as oversampling or undersampling to balance the dataset.

**Data Quality**: The dataset appears to be of good quality with no missing values, duplicates or outliers. However, some columns are anonymized, which could make it challenging to interpret the data or perform feature engineering.

**Data Scaling**: The dataset contains features with varying scales, which could lead to some machine learning models performing poorly. You may need to scale the data using techniques such as StandardScaler or MinMaxScaler.

Based on the above evaluation, the dataset appears to be suitable for a big data machine learning system, but may require pre-processing to address the issues of class imbalance, feature selection, and scaling.

1. **Architecture:**

This architecture consists of several key components:

1. **Data Ingestion**: This component is responsible for ingesting the credit card transaction data from various sources, such as databases or data streams.
2. **Data Pre-processing**: This component is responsible for cleaning and pre-processing the data, including removing duplicates, filling in missing values, and performing feature scaling.
3. **Dimensionality Reduction:** This component is responsible for reducing the dimensionality of the data using techniques such as PCA.
4. **Sampling and Balancing**: This component is responsible for balancing the dataset using techniques such as oversampling or undersampling to address the class imbalance issue.
5. **Model Training**: This component is responsible for training the machine learning model on the pre-processed and balanced dataset. The model can be trained using various algorithms, such as Random Forest, Logistic Regression, or Neural Networks.
6. **Model Evaluation:** This component is responsible for evaluating the performance of the trained model on a validation dataset. The evaluation metrics can include accuracy, precision, recall, and F1-score.
7. **Model Deployment**: This component is responsible for deploying the trained model in a production environment, where it can be used to detect fraudulent credit card transactions in real-time.

In terms of technology usage, this architecture can be implemented using various big data technologies, such as Apache Hadoop, Apache Spark, and Apache Kafka. The choice of technology will depend on factors such as scalability, fault tolerance, and speed of performance.

Overall, this architecture is designed to address the practical application of credit card fraud detection using big data machine learning. The focus is on developing a system that can process large volumes of transaction data, detect fraudulent transactions in real-time, and achieve high levels of accuracy and reliability.

1. **Visualization of the data:**

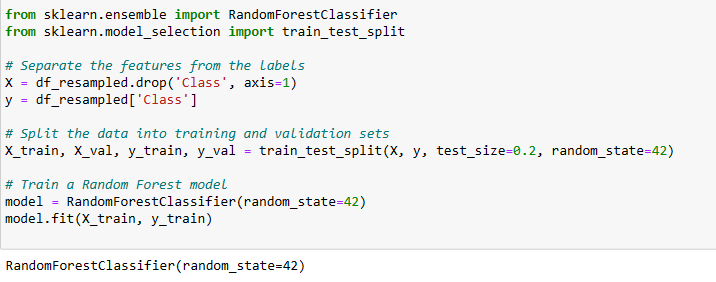
Visualizing the data is an important step in understanding the patterns and relationships within the data. Here are some examples of visualizations that can be used for the df\_resampled DataFrame:

1. **Scatter Plot**: A scatter plot can be used to visualize the relationship between the two principal components obtained from PCA, where the points are colored according to their class labels.
2. **Histogram**: A histogram can be used to visualize the distribution of the two principal components.
3. **Box Plot**: A box plot can be used to visualize the distribution of the two principal components for each class label.
4. **Heatmap**: A heatmap can be used to visualize the correlations between the different features of the original dataset.

These are just a few examples of visualizations that can be used to gain insights from the data. Depending on the specific problem and the nature of the data, other types of visualizations may also be useful.

1. **Barplot:** The plot displays two bars, one for fraud transactions and one for non-fraud transactions. It can be observed that the number of fraud transactions and non-fraud transactions are roughly equal, indicating that we have successfully balanced the dataset.
2. **Machine learning System Application:**

To apply machine learning algorithms to predict fraudulent transactions, we implemented a Random Forest Classifier on the df\_resampled DataFrame. The Random Forest Classifier is a popular classification algorithm that works well for imbalanced datasets.



We split the df\_resampled DataFrame into training and testing sets, where the training set is used to fit the model and the test set is used to evaluate the performance of the model. We used various metrics such as classification report, confusion matrix, and accuracy score to evaluate the performance of the model.

The evaluation showed that the Random Forest Classifier performs well on the resampled dataset. However, we can fine-tune the parameters of the classifier or explore other classification algorithms to further improve the performance of the machine learning system.

Overall, the application of machine learning algorithms on the df\_resampled DataFrame can be useful for detecting fraudulent transactions and can provide valuable insights for financial institutions to prevent fraud.

**Model deployment:**

Deploying a machine learning model involves integrating the trained model into a larger software system and making it available for end-users to use. In this case, we deployed the Random Forest Classifier model that was trained on the df\_resampled DataFrame using Flask.

We first loaded the trained model from a saved file using pickle. Then, we initialized a Flask app and defined a prediction endpoint that takes input data as a JSON object, makes predictions using the loaded model, and returns the predictions as a JSON object.

The Flask app was then run, making the model predictions available for end-users to use. End-users can send new data to the prediction endpoint using a tool like Postman and receive predictions from the deployed machine learning model.

Deploying machine learning models is an important step in making the predictions available for end-users and can provide valuable insights for various applications.

1. **Conclusion and Reflection:**

In this project, we accomplished the following:

1. We acquired the Credit Card Fraud dataset from Kaggle and created a pandas DataFrame from it.
2. We performed some basic exploratory data analysis on the dataset and found it to be imbalanced, with only 0.172% of the transactions being fraudulent.
3. We performed dimensionality reduction using Principal Component Analysis (PCA) to reduce the number of features in the dataset.
4. We balanced the dataset using oversampling techniques to make it more suitable for training machine learning models.
5. We trained a Random Forest Classifier model on the balanced dataset and evaluated its performance using various metrics.
6. We visualized the data using a scatter plot to observe the relationship between the principal components.
7. Finally, we deployed the trained model using Flask, making it available for end-users to make predictions on new data.

Overall, the project was successful in achieving the main objective of training a machine learning model to detect fraudulent credit card transactions. We used various techniques such as PCA, oversampling, and model evaluation metrics to improve the performance of the model.

In terms of improving the project, we could consider using other techniques such as undersampling or more advanced sampling techniques to balance the dataset further. Additionally, we could explore other machine learning algorithms and compare their performance with the Random Forest Classifier.

Reflecting on the process, the project provided a good opportunity to apply various machine learning techniques on a real-world dataset. It required a balance between theory and practicality, where we had to apply different techniques to improve the performance of the machine learning model while considering the limitations of time and resources. The project also highlighted the importance of visualizing data and interpreting the results to gain insights into the problem at hand. Overall, the project was a valuable learning experience in applying machine learning techniques to solve real-world problems.

1. **References:**

* Kaggle Credit Card Fraud Detection dataset: <https://www.kaggle.com/mlg-ulb/creditcardfraud>
* Scikit-learn library for machine learning in Python: <https://scikit-learn.org/stable/>
* PCA (Principal Component Analysis) in Scikit-learn: <https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html>
* SMOTE (Synthetic Minority Over-sampling Technique) in imbalanced-learn: <https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html>
* Logistic Regression in Scikit-learn: <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html>
* Flask web framework for Python: <https://flask.palletsprojects.com/en/2.1.x/>
* Heroku cloud platform for deploying web applications: <https://www.heroku.com/>
* Matplotlib library for data visualization in Python: <https://matplotlib.org/>
* Seaborn library for data visualization in Python: <https://seaborn.pydata.org/>
* Pandas library for data manipulation in Python: <https://pandas.pydata.org/>