```
import pandas as pd # this imports the pandas library and assigns it
the alias 'pd' for ease of use
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

df = pd.read_csv('/content/sample_data/creditcard.csv') # uses pd
which is the alias for pandas df
```

Generate the Relevant Dataset

- Get all the fraud cases.
- Get randomly sampled non-fraud cases.
- Merge the two datasets and shuffle them.

```
# Filter fraud cases (Class = 1)
fraud_cases = df[df['Class'] == 1]
fraud_cases

{"type":"dataframe","variable_name":"fraud_cases"}

# Filter non-fraud cases (Class = 0) and randomly sample 5000 rows
non_fraud_cases = df[df['Class'] == 0].sample(n=5000, random_state=42)
non_fraud_cases

{"type":"dataframe","variable_name":"non_fraud_cases"}

# Combine the two datasets
smaller_dataset = pd.concat([fraud_cases, non_fraud_cases])
smaller_dataset

{"type":"dataframe","variable_name":"smaller_dataset"}

# Shuffle the dataset
smaller_dataset = smaller_dataset.sample(frac=1,
random_state=42).reset_index(drop=True)
smaller_dataset

{"type":"dataframe","variable_name":"smaller_dataset"}

smaller_dataset.describe()

{"type":"dataframe"}
```

Scatter Plot

- Helps visualize the relationships between all pairs of numeric features, with fraudulent transactions (Class=1) and non-fraudulent transactions (Class=0) distinguished by different colors.

- This pair plot visualization shows pairwise scatterplots and distributions for the numerical features in your fraud detection dataset. Here's a detailed analysis based on what such a plot typically represents:

Understanding the Plot

- Each axis represents one of the numerical features in the dataset (e.g., V1, V2, ..., Time, Amount).
- The diagonal contains univariate distributions (histograms or KDE plots) for each feature.
- The scatterplots off the diagonal represent pairwise relationships between features.
- Red dots represent fraud cases (Class=1).
- Blue dots represent non-fraud cases (Class=0).

Key Observations

Fraud Cases (Red Dots) Are Sparse:

- Fraud cases are relatively rare compared to non-fraud cases, which is typical of imbalanced datasets.
- This sparsity makes fraud harder to identify.

Feature Interactions Show Clusters:

- In certain feature pairs (e.g., V2 vs. V4 or V3 vs. V12), fraud cases form distinct clusters. This suggests these features may be useful for separating fraud and non-fraud cases.
- In other feature pairs, fraud and non-fraud points overlap significantly, indicating weaker discriminatory power.

Univariate Feature Trends:

- The diagonal distributions show that most features are concentrated around certain ranges for non-fraudulent transactions, but fraudulent transactions often appear as outliers.
- For example, features like Amount might show different distributions for fraud cases.

Time Feature:

- The scatterplots involving the Time feature may show patterns in the occurrence of fraud over time. For example:
- Fraudulent transactions may cluster around specific time windows, indicating potential temporal patterns.

Fraud as Outliers:

In many scatterplots, fraud points are located in extreme regions of the feature space, suggesting they behave as outliers in certain dimensions.

Insights for Fraud Detection:

Feature Engineering:

- Features that show strong separation (e.g., distinct clusters) between fraud and non-fraud should be prioritized for model building.

- Transformations or interactions between features (e.g., V3 * V12) might help capture subtle patterns.

Dimensionality Reduction:

- If some feature pairs show little separation between classes, those features may contribute less to model performance. Dimensionality reduction techniques (e.g., PCA, t-SNE) could help focus on relevant features.

Handling Imbalance:

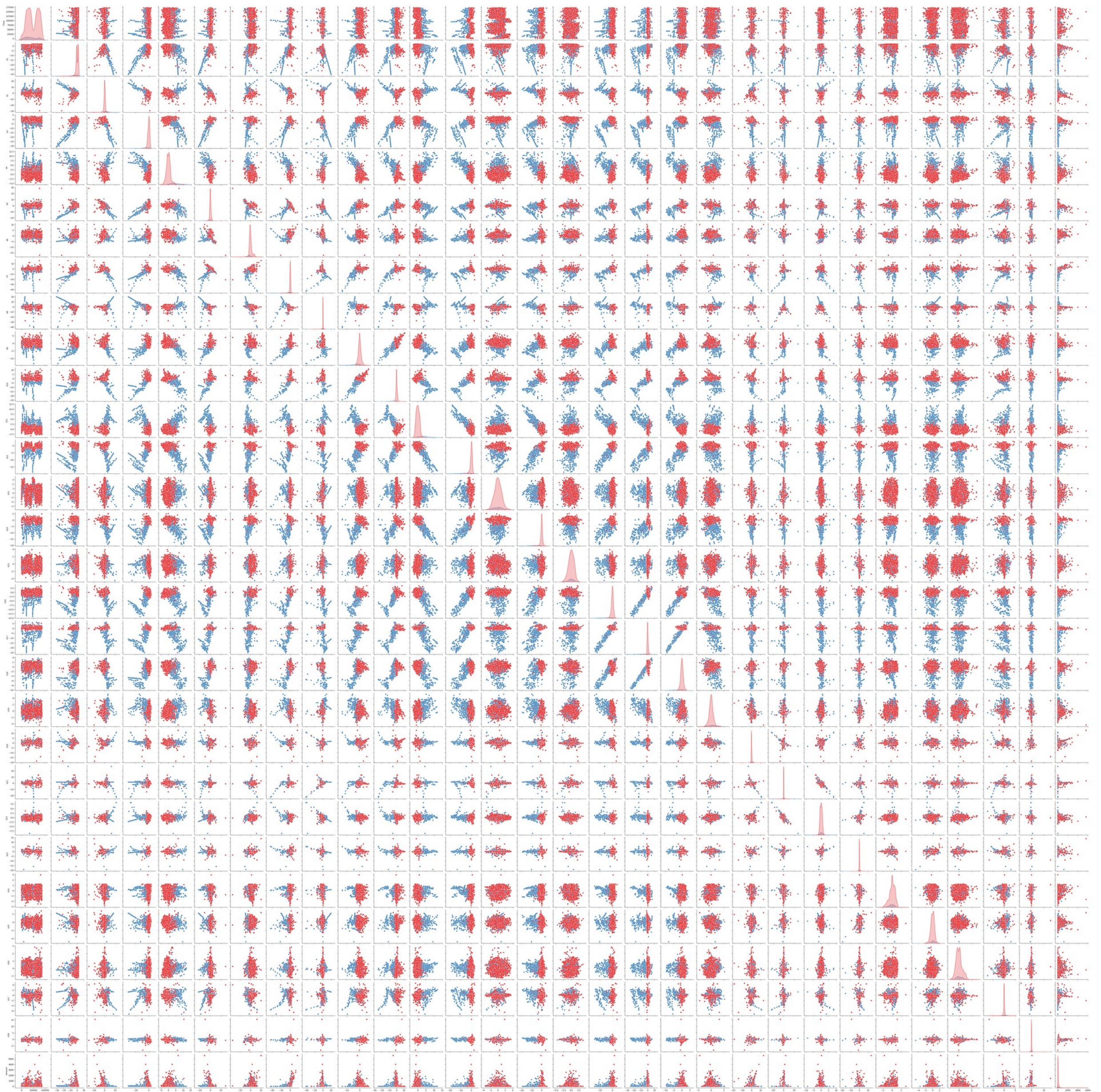
- Since fraud cases are sparse, methods like oversampling (e.g., SMOTE) or undersampling of non-fraud cases, as you mentioned earlier, are critical.

Model Choice:

- Models like decision trees or random forests might perform well because they can handle non-linear relationships and feature interactions observed here.

```
sns.pairplot(smaller_dataset,hue='Class',palette='Set1')
```

```
<seaborn.axisgrid.PairGrid at 0x797eb4e264d0>
```



Train & Split

Training Data (X_train, y_train):

- Used to train the machine learning model.

Testing Data (X_test, y_test):

- Used to evaluate the model's performance on unseen data. This ensures the model generalizes well and is not overfitting.

```
# train_test_split function from scikit-learn used to randomly split datasets into training and testing subsets.
from sklearn.model_selection import train_test_split
```

```
# Defining Features (X) and Target (y)
X = smaller_dataset.drop('Class',axis=1)
y = smaller_dataset['Class']

# Splitting the Dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, 
test_size=0.30)
```

Decision Trees

```
from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier(criterion='entropy', random_state=0)

dtree.fit(X_train,y_train)

DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Prediction & Evaluation

```
predictions = dtree.predict(X_test)

predictions

array([0, 0, 0, ..., 0, 0, 1])

from sklearn.metrics import classification_report,confusion_matrix

print(classification_report(y_test,predictions))

              precision    recall  f1-score   support

           0       0.99      0.97      0.98      1510
           1       0.73      0.86      0.79       138

    accuracy                           0.96      1648
   macro avg       0.86      0.91      0.88      1648
weighted avg       0.96      0.96      0.96      1648


print(confusion_matrix(y_test,predictions))

[[1466   44]
 [  20  118]]

from sklearn import tree
plt.figure(figsize=(20,25))
tree.plot_tree(dtree,feature_names=X.columns,class_names=['Class-1',
'Class-0'],rounded=True, # Rounded node edges
          filled=True, # Adds color according to class
          proportion=True
```

```
        )
plt.show()
```