**Credit Card Fraud Detection**

Detecting credit card fraud is a critical application of applied data science. To iovate and improve a credit card fraud detection project, you can follow these steps

1. **Data Collection:**

- Gather comprehensive transaction data, including both legitimate and fraudulent transactions.

- Obtain historical data that represents different patterns of transactions.

**2. Data Preprocessing:**

- Clean and preprocess the data to handle missing values, outliers, and noise.

- Normalize or standardize features to ensure consistency.

3**. Feature Engineering**:

- Create relevant features from the data that can help the model differentiate between legitimate and fraudulent transactions.

- Consider using techniques like PCA (Principal Component Analysis) to reduce dimensionality.

**4. Data Split:**

- Divide the data into training, validation, and test sets to evaluate your model's performance accurately.

**5. Model Selection and Development:**

- Explore various machine learning algorithms, such as logistic regression, decision trees, random forests, and neural networks.

- Experiment with ensemble methods and deep learning architectures for improved accuracy.

**6. Hyperparameter Tuning:**

- Fine-tune the hyperparameters of your chosen models to optimize their performance.

- Use techniques like grid search or random search to identify the best hyperparameters.

**7. Imbalanced Data Handling:**

- Since fraud cases are typically rare, employ techniques like oversampling, undersampling, or synthetic data generation (SMOTE) to address class imbalance.

**8. Evaluation Metrics**:

- Use appropriate evaluation metrics like precision, recall, F1-score, and AUC-ROC to assess model performance.

- Consider the business impact of false positives and false negatives when setting thresholds.

**9. Cross-Validation:**

- Implement cross-validation to assess the model's generalization ability and mitigate overfitting.

10. Real-time Monitoring:

- Deploy the model in a real-time environment where it can continuously monitor and detect fraudulent transactions.

**11. Anomaly Detection:**

- Combine traditional supervised learning with unsupervised anomaly detection methods, such as isolation forests, to identify novel fraud patterns.

**12. Explainability**:

- Ensure that your model provides interpretable explanations for its predictions, which can be crucial for regulatory compliance and trust.

**13. Continuous Learning:**

- Continuously update and retrain your model as new data becomes available to adapt to evolving fraud patterns.

**14. Collaboration:**

- Work closely with domain experts, risk analysts, and fraud investigators to gain insights and improve model accuracy.

**15. Security:**

- Implement robust security measures to protect sensitive transaction data and the model itself from potential attacks.

**16. Regulatory Compliance:**

- Ensure that your project complies with relevant data protection and financial industry regulations, such as GDPR and PCI DSS.

**17. Documentation:**

- Maintain thorough documentation of the project, including data sources, preprocessing steps, model architecture, and results.

**18. Ethical Considerations:**

- Address ethical concerns related to data privacy and fairness, and implement safeguards against bias and discrimination.

**19. Reporting:**

- Generate regular reports and alerts for relevant stakeholders, including financial institutions, about detected fraud incidents and model performance.

**20. User Education:**

- Educate users and customers about best practices for securing their credit cards and recognizing potential fraud.in credit card fraud detection requires a combination of advanced machine learning techniques, domain expertise, and a commitment to staying up-to-date with evolving fraud tactics and data security practices.

## **Exploratory Analysis**

To begin this exploratory analysis, first use matplotlib to import libraries and define functions for plotting the data. Depending on the data, not all plots will be made.

from mpl\_toolkits.mplot3d import Axes3D

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt *# plotting*

import numpy as np *# linear algebra*

import os *# accessing directory structure*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

There is 1 csv file in the current version of the dataset:

In [2]:

print(os.listdir('../input'))

['creditcard.csv']

linkcode

The next hidden code cells define functions for plotting data. Click on the "Code" button in the published kernel to reveal the hidden code.

*Distribution graphs (histogram/bar graph) of column data*

def plotPerColumnDistribution(df, nGraphShown, nGraphPerRow):

nunique = df.nunique()

df = df[[col for col **in** df if nunique[col] > 1 **and** nunique[col] < 50]] *# For displaying purposes, pick columns that have between 1 and 50 unique values*

nRow, nCol = df.shape

columnNames = list(df)

nGraphRow = (nCol + nGraphPerRow - 1) / nGraphPerRow

plt.figure(num = None, figsize = (6 \* nGraphPerRow, 8 \* nGraphRow), dpi = 80, facecolor = 'w', edgecolor = 'k')

for i **in** range(min(nCol, nGraphShown)):

plt.subplot(nGraphRow, nGraphPerRow, i + 1)

columnDf = df.iloc[:, i]

if (**not** np.issubdtype(type(columnDf.iloc[0]), np.number)):

valueCounts = columnDf.value\_counts()

valueCounts.plot.bar()

else:

columnDf.hist()

plt.ylabel('counts')

plt.xticks(rotation = 90)

plt.title(f'**{columnNames[i]}** (column **{i}**)')

plt.tight\_layout(pad = 1.0, w\_pad = 1.0, h\_pad = 1.0)

plt.show()

*Correlation matrix*

def plotCorrelationMatrix(df, graphWidth):

filename = df.dataframeName

df = df.dropna('columns') *# drop columns with NaN*

df = df[[col for col **in** df if df[col].nunique() > 1]] *# keep columns where there are more than 1 unique values*

if df.shape[1] < 2:

print(f'No correlation plots shown: The number of non-NaN or constant columns (**{df.shape[1]}**) is less than 2')

return

corr = df.corr()

plt.figure(num=None, figsize=(graphWidth, graphWidth), dpi=80, facecolor='w', edgecolor='k')

corrMat = plt.matshow(corr, fignum = 1)

plt.xticks(range(len(corr.columns)), corr.columns, rotation=90)

plt.yticks(range(len(corr.columns)), corr.columns)

plt.gca().xaxis.tick\_bottom()

plt.colorbar(corrMat)

plt.title(f'Correlation Matrix for **{filename}**', fontsize=15)

plt.show()

*# Scatter and density plots*

def plotScatterMatrix(df, plotSize, textSize):

df = df.select\_dtypes(include =[np.number]) *# keep only numerical columns*

*# Remove rows and columns that would lead to df being singular*

df = df.dropna('columns')

df = df[[col for col **in** df if df[col].nunique() > 1]] *# keep columns where there are more than 1 unique values*

columnNames = list(df)

if len(columnNames) > 10: *# reduce the number of columns for matrix inversion of kernel density plots*

columnNames = columnNames[:10]

df = df[columnNames]

ax = pd.plotting.scatter\_matrix(df, alpha=0.75, figsize=[plotSize, plotSize], diagonal='kde')

corrs = df.corr().values

for i, j **in** zip(\*plt.np.triu\_indices\_from(ax, k = 1)):

ax[i, j].annotate('Corr. coef = **%.3f**' % corrs[i, j], (0.8, 0.2), xycoords='axes fraction', ha='center', va='center', size=textSize)

plt.suptitle('Scatter and Density Plot')

plt.show()

Now you're ready to read in the data and use the plotting functions to visualize the data.

### **Let's check 1st file: ../input/creditcard.csv**

In [6]:

nRowsRead = 1000 *# specify 'None' if want to read whole file*

*# creditcard.csv has 284807 rows in reality, but we are only loading/previewing the first 1000 rows*

df1 = pd.read\_csv('../input/creditcard.csv', delimiter=',', nrows = nRowsRead)

df1.dataframeName = 'creditcard.csv'

nRow, nCol = df1.shape

print(f'There are **{nRow}** rows and **{nCol}** columns')

There are 1000 rows and 31 columns

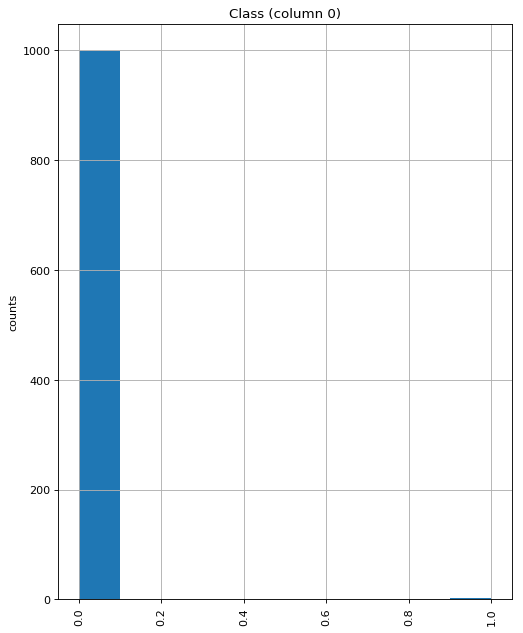
linkcode

Let's take a quick look at what the data looks like:

df1.head(5) df1.head(5)

<https://www.kaggle.com/code/varnika777/starter-credit-card-fraud-detection-2cb0c438-f?scriptVersionId=11343240&cellId=14>

Distribution graphs (histogram/bar graph) of sampled columns:

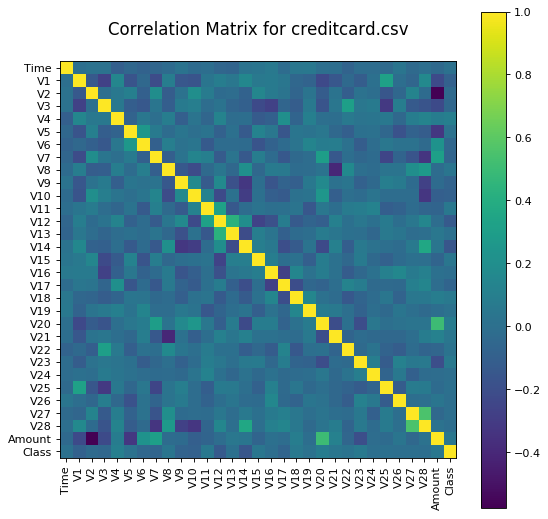
In [8]:

Correlation matrix:

In [9]:

linkcode

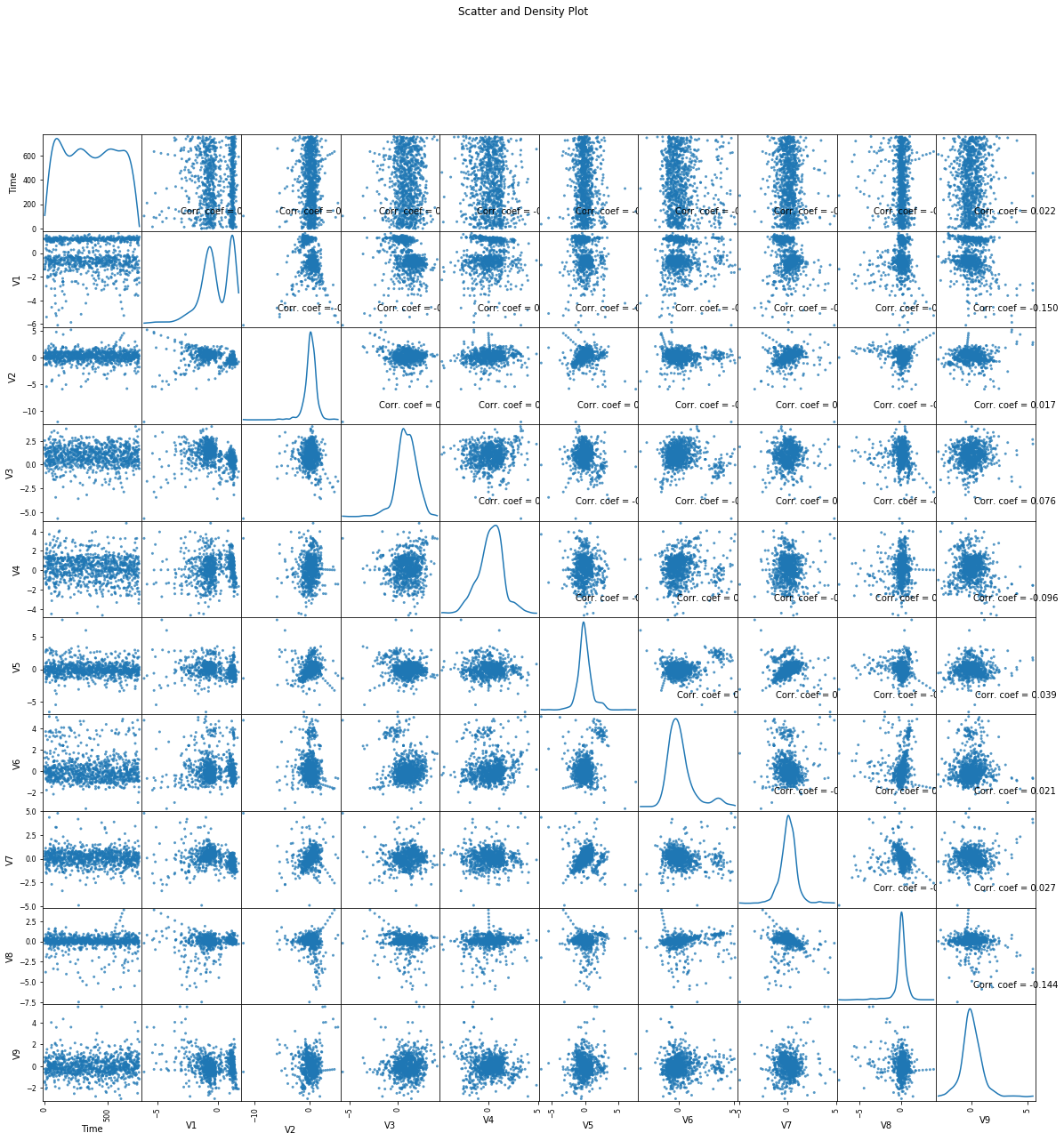
plotCorrelationMatrix(df1, 8)



Scatter and density plots:

In [10]:

plotScatterMatrix(df1, 20, 10)



**DONE BY:**

**Jeripiti Venkata Tarun**

**REG NO:720921244023**