Credit Card Fraud Detection

Importing the libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PowerTransformer
from imblearn.over_sampling import SMOTE
from collections import Counter
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
%matplotlib inline
```

Importing the dataset

```
df = pd.read_csv('/content/creditcard.csv')
df.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321	-1.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458	0.

5 rows × 31 columns

Checking the discrepencies in the data and performing exploratory data analysis

```
df.isna().sum()
```

```
Time      0
V1        0
V2        0
V3        0
V4        0
V5        0
V6        0
V7        0
V8        0
V9        0
V10       0
V11       0
V12       0
V13       0
V14       0
V15       0
V16       0
V17       0
V18       0
V19       0
V20       0
V21       0
V22       0
V23       0
V24       0
```

```
V25       0
V26       0
V27       0
V28       0
Amount    0
Class     0
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 284807 entries, 0 to 284806
Data columns (total 31 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   Time    284807 non-null  float64
 1   V1      284807 non-null  float64
 2   V2      284807 non-null  float64
 3   V3      284807 non-null  float64
 4   V4      284807 non-null  float64
 5   V5      284807 non-null  float64
 6   V6      284807 non-null  float64
 7   V7      284807 non-null  float64
 8   V8      284807 non-null  float64
 9   V9      284807 non-null  float64
 10  V10     284807 non-null  float64
 11  V11     284807 non-null  float64
 12  V12     284807 non-null  float64
 13  V13     284807 non-null  float64
 14  V14     284807 non-null  float64
 15  V15     284807 non-null  float64
 16  V16     284807 non-null  float64
 17  V17     284807 non-null  float64
 18  V18     284807 non-null  float64
 19  V19     284807 non-null  float64
 20  V20     284807 non-null  float64
 21  V21     284807 non-null  float64
 22  V22     284807 non-null  float64
 23  V23     284807 non-null  float64
 24  V24     284807 non-null  float64
 25  V25     284807 non-null  float64
 26  V26     284807 non-null  float64
 27  V27     284807 non-null  float64
 28  V28     284807 non-null  float64
 29  Amount  284807 non-null  float64
 30  Class   284807 non-null  int64
dtypes: float64(30), int64(1)
memory usage: 67.4 MB
```

```
len(df)
```

```
284807
```

```
df.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.433583e-02	-2.741871e-01	4.010308e-02	2.235804e-02	-5
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.119264e-01	3.985649e-01	5.704361e-01	3.273459e-01	5
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.480167e+01	7.330163e+01	1.205895e+02	2.000721e+01	1

8 rows × 31 columns

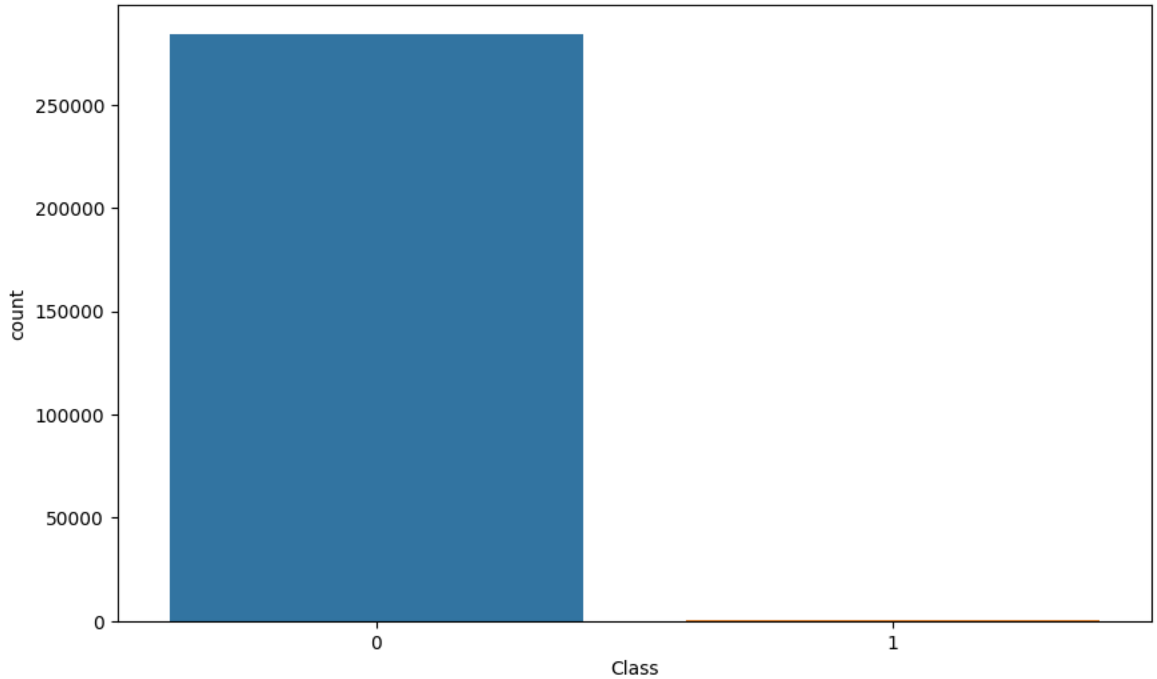
```
# The classes are heavily skewed we need to solve this issue later.
print('No Frauds', round(df['Class'].value_counts()[0]/len(df) * 100,2), '% of the dataset')
print('Frauds', round(df['Class'].value_counts()[1]/len(df) * 100,2), '% of the dataset')
```

```
No Frauds 99.83 % of the dataset
Frauds 0.17 % of the dataset
```

Checking the distribution of data

```
plt.figure(dpi=100, figsize=(10,6))
sns.countplot(data=df, x='Class')
```

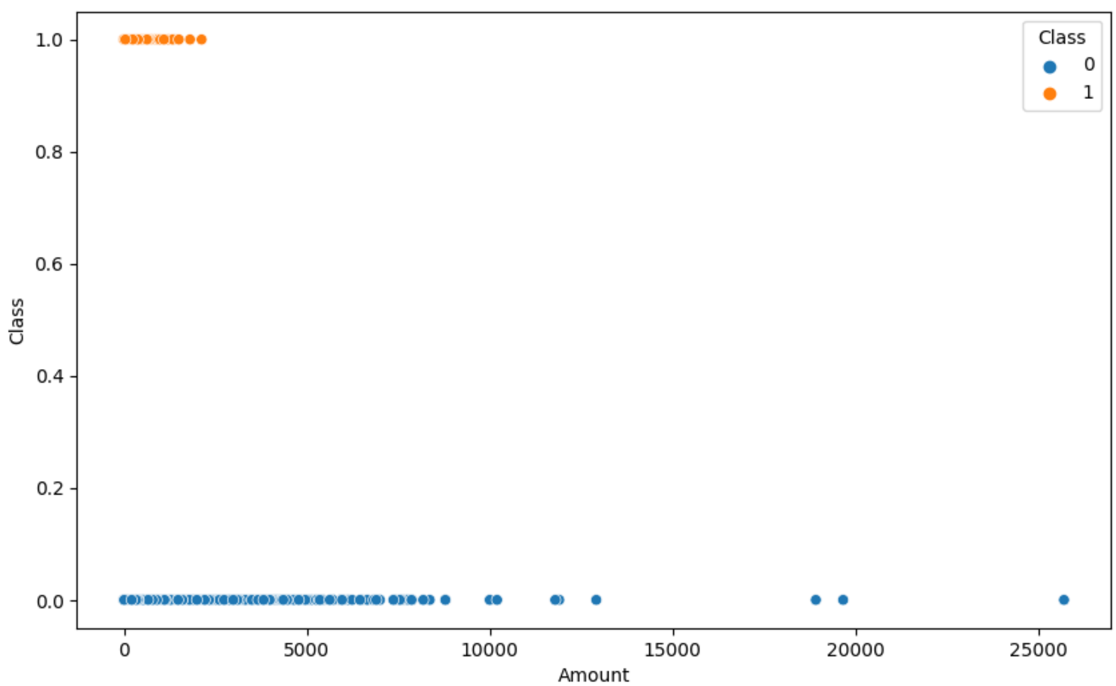
```
<Axes: xlabel='Class', ylabel='count'>
```



Checking the effect of `Amount` and `Time` columns of the dataset on `Class`

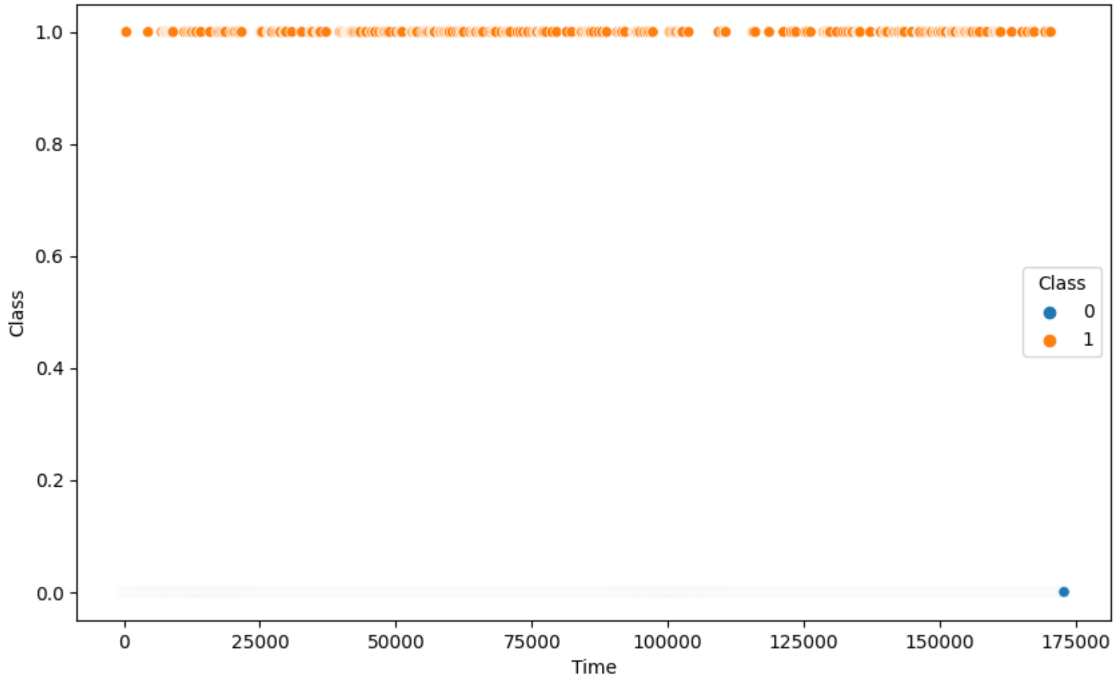
```
plt.figure(dpi=100, figsize=(10,6))
sns.scatterplot(data=df, x='Amount', y='Class', hue='Class')
```

```
<Axes: xlabel='Amount', ylabel='Class'>
```



```
plt.figure(dpi=100, figsize=(10,6))
sns.scatterplot(data=df, x='Time', y='Class', hue='Class', )
```

```
<Axes: xlabel='Time', ylabel='Class'>
```



Dropping the non impactful columns

```
df = df.drop(['Time'], axis=1)
df.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458

5 rows × 30 columns

Scaling `Amount` feature for better results

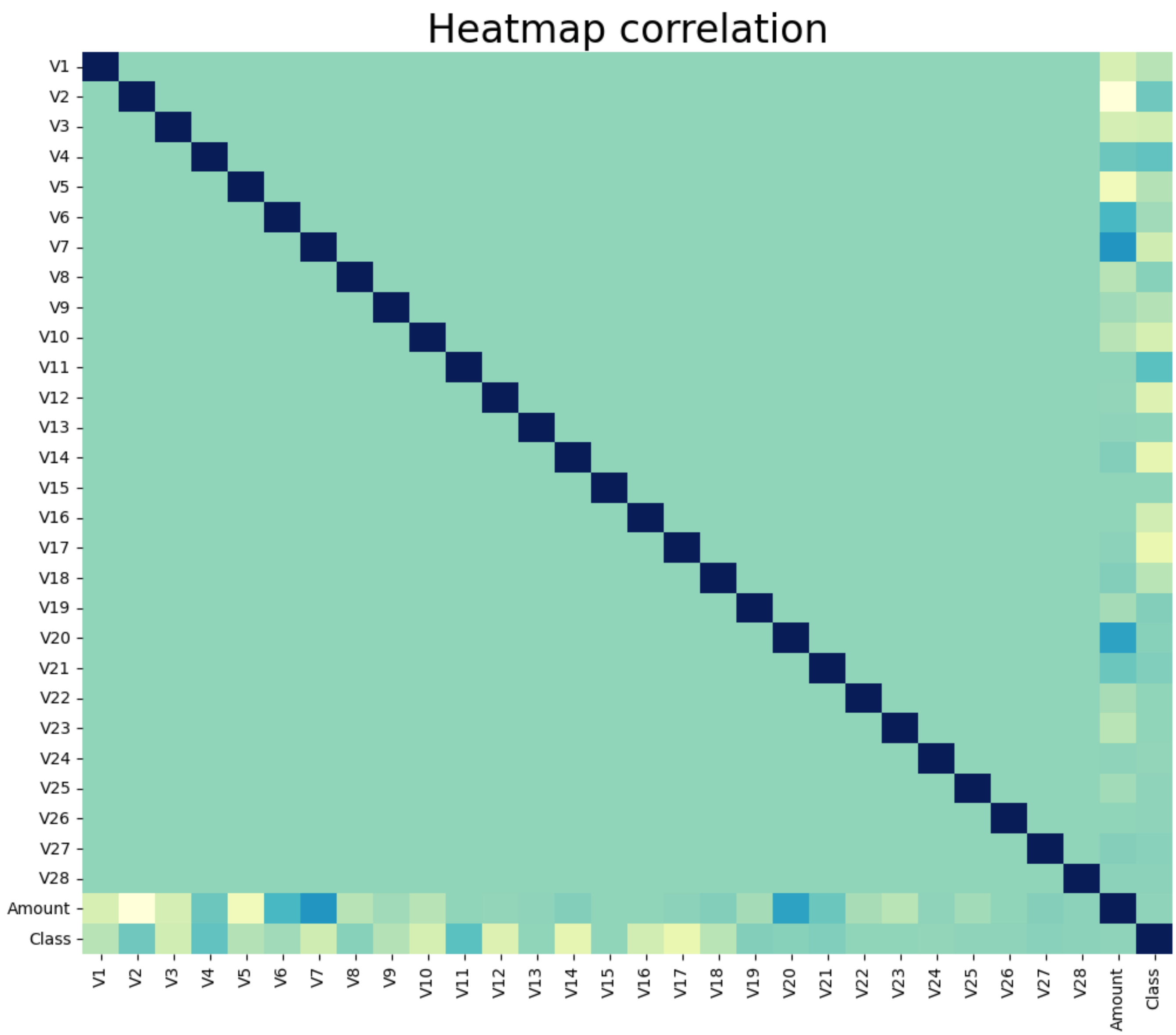
```
sc = StandardScaler()
amount = df['Amount'].values

df['Amount'] = sc.fit_transform(amount.reshape(-1, 1))
df.head()
```

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	...	V21	V22	V23
0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	0.090794	...	-0.018307	0.277838	-0.110474
1	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	-0.166974	...	-0.225775	-0.638672	0.101288
2	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	0.207643	...	0.247998	0.771679	0.909412
3	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	-0.054952	...	-0.108300	0.005274	-0.190321
4	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	0.753074	...	-0.009431	0.798278	-0.137458

5 rows × 30 columns

Checking the Distribution of various features of our dataset

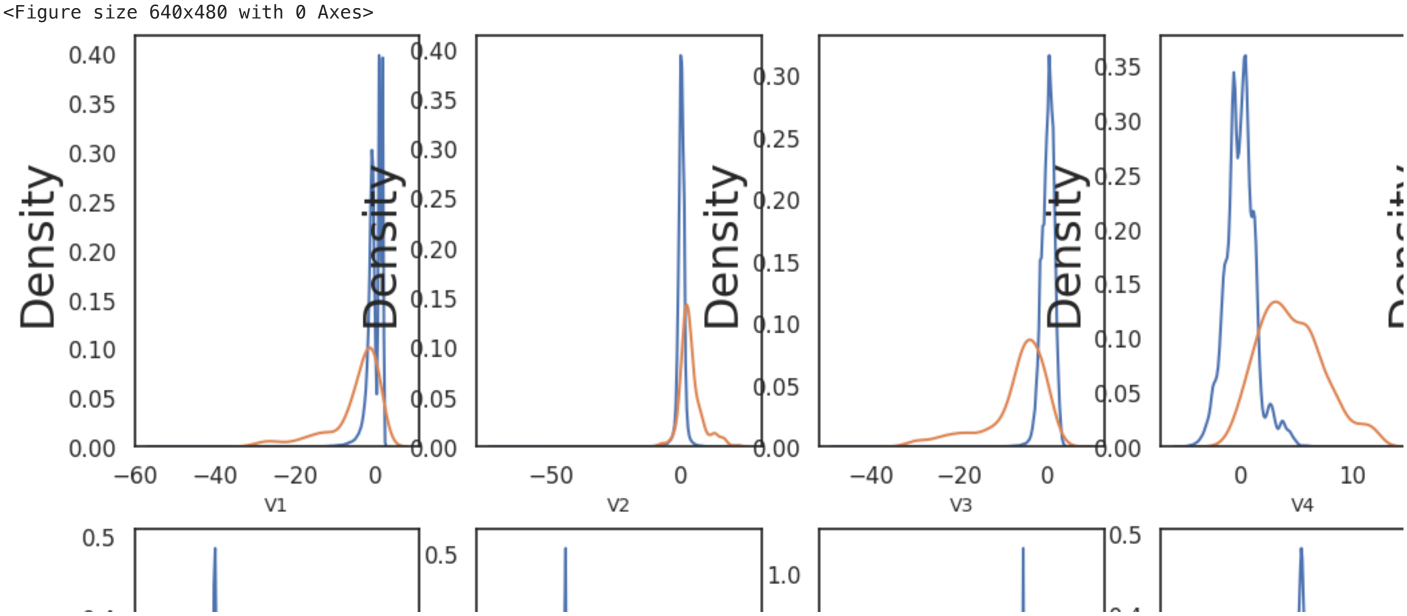
```
df_corr = df.corr() # Calculation of the correlation coefficients in pairs,
plt.figure(figsize=(15,10))
sns.heatmap(df_corr, cmap="YlGnBu") # Displaying the Heatmap
sns.set(font_scale=2,style='white')

plt.title('Heatmap correlation')
plt.show()
```

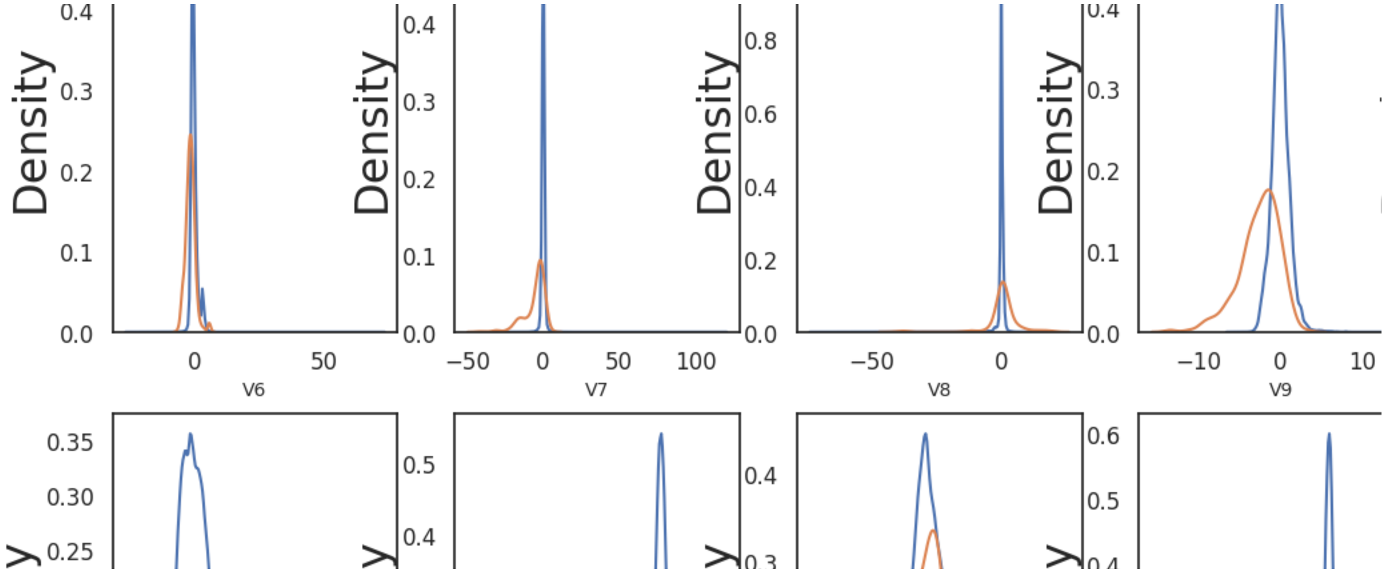



```
columns = list(df.columns.values)
columns.remove("Class")
n = 1
t0 = df.loc[df['Class'] == 0]
t1 = df.loc[df['Class'] == 1]

plt.figure()
fig, ax = plt.subplots(12,7,figsize=(16,28))

for i in columns:
    plt.subplot(6,5,n)
    sns.kdeplot(t0[i],label="0")
    sns.kdeplot(t1[i],label="1")
    plt.xlabel(i, fontsize=10)
    locs, labels = plt.xticks()
    plt.tick_params(axis='both', which='major', labelsize=12)
```

```
    n =n + 1
plt.show();
```

```
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<ipython-input-34-e30f830577e9>:11: MatplotlibDeprecationWarning: Auto-removal of overlapping axes is deprecated since 3.6 ar
  plt.subplot(6,5,n)
<Figure size 640x480 with 0 Axes>
```





Preparing the dataset for training

```
X = df.iloc[:, :-1]
y = df.iloc[:, -1]
```

Splitting the dataset into training and validation sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=7, test_size=0.4)
len(X_train)
```

```
170884
```

Creating synthetic data using `SMOTE` (Since the dataset is imbalanced)

```
smote = SMOTE()
X_train_smote, y_train_smote = smote.fit_resample(X_train.astype('float'), y_train)
print("Before performing smote : ", Counter(y_train))
print("After performing smote : ", Counter(y_train_smote))
```

```
Before performing smote :  Counter({0: 170580, 1: 304})
After performing smote :  Counter({0: 170580, 1: 170580})
```

```
from sklearn.preprocessing import StandardScaler
# Create a StandardScaler object
scaler = StandardScaler() #check if it already save the mean, and std

# Fit and transform the scaler on the training data
X_train_smote = scaler.fit_transform(X_train_smote)

# Use the same mean and std to transform the testing data
X_test = scaler.transform(X_test)
```

```
print("X_train - ",X_train.shape)
print("y_train - ",y_train.shape)
print("X_train_smote - ",X_train_smote.shape)
print("y_train_smote - ",y_train_smote.shape)
print("X_test - ",X_test.shape)
print("y_test - ",y_test.shape)
```

```
X_train -  (170884, 29)
y_train -  (170884,)
X_train_smote -  (341160, 29)
y_train_smote -  (341160,)
X_test -  (113923, 29)
y_test -  (113923,)
```

Testing various models on the dataset

1.1. Logistic Regression without synthetic data

```
model_ws_1 = LogisticRegression(solver='lbfgs', max_iter=1000)
model_ws_1.fit(X_train, y_train)
y_pred_ws_1 = model_ws_1.predict(X_test)
acc_ws_1 = accuracy_score(y_test, y_pred_ws_1)
acc_ws_1
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticR
  warnings.warn(
0.9983936518525671
```

Logistic Regression CV without synthetic data

```
from sklearn.linear_model import LogisticRegressionCV
from sklearn.metrics import classification_report, mean_squared_error
from sklearn.model_selection import train_test_split, KFold
model_ws_1b = LogisticRegressionCV(Cs=30, cv=10, penalty="l1", n_jobs=8, max_iter=1000, solver="liblinear")
fit = model_ws_1b.fit(X_train, y_train)

y_pred_ws_1b = model_ws_1b.predict(X_test)
acc_ws_1b = accuracy_score(y_test, y_pred_ws_1)
acc_ws_1b
```

```
/usr/local/lib/python3.10/dist-packages/joblib/externals/loky/process_executor.py:752: UserWarning: A worker stopped while s
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but LogisticR
  warnings.warn(
0.9983936518525671
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
# Prepare data (X, Y), build a model, and predict into Yh
res11 = accuracy_score(y_test, y_pred_ws_1b)

print("\n", "Accuracy: ".format(format(res11,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_1b))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_1b))
```

```
 Accuracy: 

 CFM: 
 [[113731      4]
 [   180      8]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       0.67      0.04      0.08       188

    accuracy                           1.00    113923
   macro avg       0.83      0.52      0.54    113923
weighted avg       1.00      1.00      1.00    113923
```

1.2 Logistic Regression with synthetic data

```
model_s_1 = LogisticRegression(solver='lbfgs', max_iter=1000)
model_s_1.fit(X_train_smote, y_train_smote)
y_pred_s_1 = model_s_1.predict(X_test)
acc_s_1 = accuracy_score(y_test, y_pred_s_1)
acc_s_1
```

```
0.9725955250476199
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res12 = accuracy_score(y_test, y_pred_s_1)

print("\n", "Accuracy: ".format(format(res12,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_s_1))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_s_1))
```

```
 Accuracy:

 CFM:
 [[110631   3104]
 [    18    170]]

 Classification report:
               precision    recall  f1-score   support

           0       1.00      0.97      0.99    113735
           1       0.05      0.90      0.10       188

    accuracy                           0.97    113923
   macro avg       0.53      0.94      0.54    113923
weighted avg       1.00      0.97      0.98    113923
```

2.1. Decision Tree Classifier without synthetic data

```
model_ws_2 = DecisionTreeClassifier(criterion='entropy', max_depth=5)
model_ws_2.fit(X_train, y_train)
y_pred_ws_2 = model_ws_2.predict(X_test)
acc_ws_2 = accuracy_score(y_test, y_pred_ws_2)
acc_ws_2
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but DecisionT
  warnings.warn(
0.9983936518525671
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

Confusion Matrix,

```
res21 = accuracy_score(y_test, y_pred_ws_2)

print("\n", "Accuracy: ".format(format(res21,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_2))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_2))
```

```
 Accuracy:

 CFM:
 [[113735      0]
 [   183      5]]

 Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       1.00      0.03      0.05       188

    accuracy                           1.00    113923
   macro avg       1.00      0.51      0.53    113923
weighted avg       1.00      1.00      1.00    113923
```

2.2. Decision Tree Classifier with synthetic data

```
model_s_2 = DecisionTreeClassifier(criterion='entropy', max_depth=5)
model_s_2.fit(X_train_smote, y_train_smote)
y_pred_s_2 = model_s_2.predict(X_test)
```

```
acc_s_2 = accuracy_score(y_test, y_pred_s_2)
acc_s_2
```

```
0.9429439182605795
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res22 = accuracy_score(y_test, y_pred_s_2)

print("\n", "Accuracy: ".format(format(res22,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_s_2))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_s_2))
```

```
 Accuracy: 

 CFM: 
 [[107248   6487]
 [    13    175]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      0.94      0.97    113735
           1       0.03      0.93      0.05       188

    accuracy                           0.94    113923
   macro avg       0.51      0.94      0.51    113923
weighted avg       1.00      0.94      0.97    113923
```

3.1. K Nearest Neighbors Classifier without synthetic data

```
model_ws_3 = KNeighborsClassifier(n_neighbors=3)
model_ws_3.fit(X_train, y_train)
y_pred_ws_3 = model_ws_3.predict(X_test)
acc_ws_3 = accuracy_score(y_test, y_pred_ws_3)
acc_ws_3
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but KNeighbor
  warnings.warn(
0.9984902082985876
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res31 = accuracy_score(y_test, y_pred_ws_3)

print("\n", "Accuracy: ".format(format(res31,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_3))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_3))
```

```
 Accuracy: 

 CFM: 
 [[113735      0]
 [   172     16]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       1.00      0.09      0.16       188

    accuracy                           1.00    113923
   macro avg       1.00      0.54      0.58    113923
weighted avg       1.00      1.00      1.00    113923
```

3.2. K Nearest Neighbors Classifier with synthetic data

```
model_s_3 = KNeighborsClassifier(n_neighbors=3)
model_s_3.fit(X_train_smote, y_train_smote)
y_pred_s_3 = model_s_3.predict(X_test)
acc_s_3 = accuracy_score(y_test, y_pred_s_3)
acc_s_3
```

```
0.9974105316749032
```

Accuracy, f1Score, precision and recall of the model

```
res32 = accuracy_score(y_test, y_pred_s_3)

print("\n", "Accuracy: ".format(format(res32,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_s_3))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_s_3))
```

```
 Accuracy: 

 CFM: 
 [[113470    265]
 [    30    158]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       0.37      0.84      0.52       188

    accuracy                           1.00    113923
   macro avg       0.69      0.92      0.76    113923
weighted avg       1.00      1.00      1.00    113923
```

4.1. Random Forest Classifier without synthetic data

```
model_ws_5 = RandomForestClassifier(max_depth=5, criterion='entropy')
model_ws_5.fit(X_train, y_train)
y_pred_ws_5 = model_ws_5.predict(X_test)
acc_ws_5 = accuracy_score(y_test, y_pred_ws_5)
acc_ws_5
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but RandomFor
  warnings.warn(
0.9983497625589214
```

Accuracy, f1Score, precision and recall of the model

```
res41 = accuracy_score(y_test, y_pred_ws_5)

print("\n", "Accuracy: ".format(format(res41,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_5))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_5))
```

```
 Accuracy: 

 CFM: 
 [[113735      0]
 [   188      0]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       0.00      0.00      0.00       188

    accuracy                           1.00    113923
   macro avg       0.50      0.50      0.50    113923
weighted avg       1.00      1.00      1.00    113923

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
```

```
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-sco
  _warn_prf(average, modifier, msg_start, len(result))
```

4.2. Random Forest Classifier with synthetic data

```
model_s_5 = RandomForestClassifier(max_depth=5, criterion='entropy')
model_s_5.fit(X_train_smote, y_train_smote)
y_pred_s_5 = model_s_5.predict(X_test)
acc_s_5 = accuracy_score(y_test, y_pred_s_5)
acc_s_5
```

```
0.993618496703914
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res42 = accuracy_score(y_test, y_pred_s_5)

print("\n", "Accuracy: ".format(format(res42,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_s_5))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_s_5))
```

```
 Accuracy:

 CFM:
 [[113028    707]
 [    20    168]]

 Classification report:
               precision    recall  f1-score   support

           0       1.00      0.99      1.00    113735
           1       0.19      0.89      0.32       188

    accuracy                           0.99    113923
   macro avg       0.60      0.94      0.66    113923
weighted avg       1.00      0.99      1.00    113923
```

5.1. Support Vector Classifier without synthetic data

```
model_ws_6 = SVC()
model_ws_6.fit(X_train, y_train)
y_pred_ws_6 = model_ws_6.predict(X_test)
acc_ws_6 = accuracy_score(y_test, y_pred_ws_6)
acc_ws_6
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does not have valid feature names, but SVC was f
  warnings.warn(
0.9983673182763797
```

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res51 = accuracy_score(y_test, y_pred_ws_6)

print("\n", "Accuracy: ".format(format(res51,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_6))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_6))
```

```
 Accuracy:

 CFM:
 [[113735      0]
 [   186      2]]

 Classification report:
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       1.00      0.01      0.02       188
```

```
    accuracy                           1.00    113923
   macro avg       1.00      0.51      0.51    113923
weighted avg       1.00      1.00      1.00    113923
```

5.2. Support Vector Classifier with synthetic data

```
model_s_6 = SVC()
model_s_6.fit(X_train_smote, y_train_smote)
y_pred_s_6 = model_s_6.predict(X_test)
acc_s_6 = accuracy_score(y_test, y_pred_s_6)
acc_s_6
```

```
0.9921701500136056
```

Confusion Matrix

`Accuracy`, `f1Score`, `precision` and `recall` of the model

```
res52 = accuracy_score(y_test, y_pred_ws_6)

print("\n", "Accuracy: ".format(format(res52,'.3f')))
print("\n", "CFM: \n", confusion_matrix(y_test, y_pred_ws_6))
print("\n", "Classification report: \n", classification_report(y_test, y_pred_ws_6))
```

```
 Accuracy: 

 CFM: 
 [[113735      0]
 [   186      2]]

 Classification report: 
               precision    recall  f1-score   support

           0       1.00      1.00      1.00    113735
           1       1.00      0.01      0.02       188

    accuracy                           1.00    113923
   macro avg       1.00      0.51      0.51    113923
weighted avg       1.00      1.00      1.00    113923
```

Comparing precision, accuracy f1 score and recall of all the models

```
dp = pd.DataFrame([res11,res12,res21,res22,res31,res32,res41,res42,res51,res52],index=['1.1','1.2','2.1','2.2','3.1','3.2','4.1'
```

```
dp
```

	0
1.1	0.998385
1.2	0.972596
2.1	0.998394
2.2	0.942944
3.1	0.998490
3.2	0.997411
4.1	0.998350
4.2	0.993618
5.1	0.998367
5.2	0.998367