Credit Card Fraud Detection-Support Vector Machines

Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

upload data

```
# loading the dataset to a Pandas DataFrame
credit_card_data = pd.read_csv('creditcard.csv')
```

```
credit_card_data.keys()
```

```
Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
       'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
       'Class'],
      dtype='object')
```

```
# first 5 rows of the dataset
credit_card_data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533

5 rows × 31 columns

```
credit_card_data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006

5 rows × 31 columns

```
# dataset informations
credit_card_data.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
```

```
# checking the number of missing values in each column
credit_card_data.isnull().sum()
```

```
        0
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
```

```python
# distribution of legit transactions & fraudulent transactions
credit_card_data['Class'].value_counts()
```

```
         count
Class
    0   284315
    1      492

dtype: int64
```

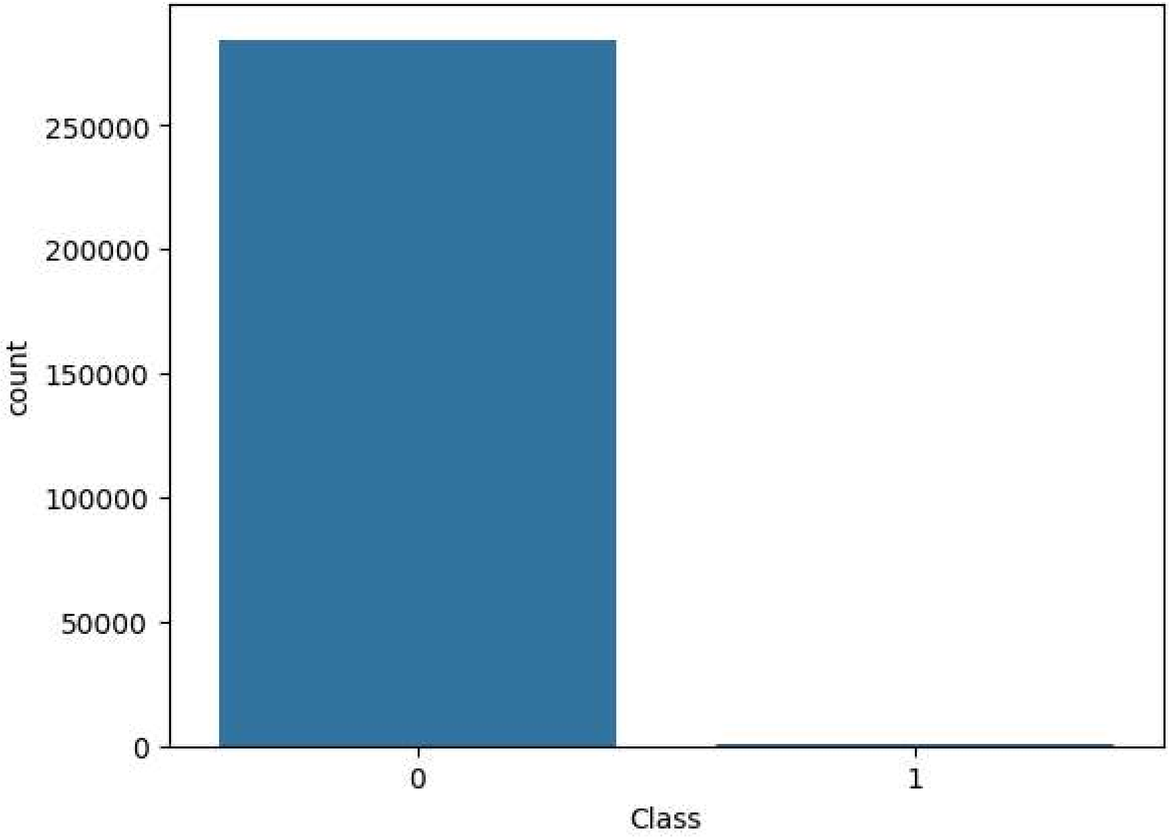
```python
credit_card_data = credit_card_data.drop("Time", axis=1)
```

```python
from sklearn import preprocessing
scaler = preprocessing.StandardScaler()
```

```python
#standard scaling
credit_card_data['std_Amount'] = scaler.fit_transform(credit_card_data['Amount'].values.resh
#removing Amount
credit_card_data = credit_card_data.drop("Amount", axis=1)
```

```python
sns.countplot(x="Class", data=credit_card_data)
```

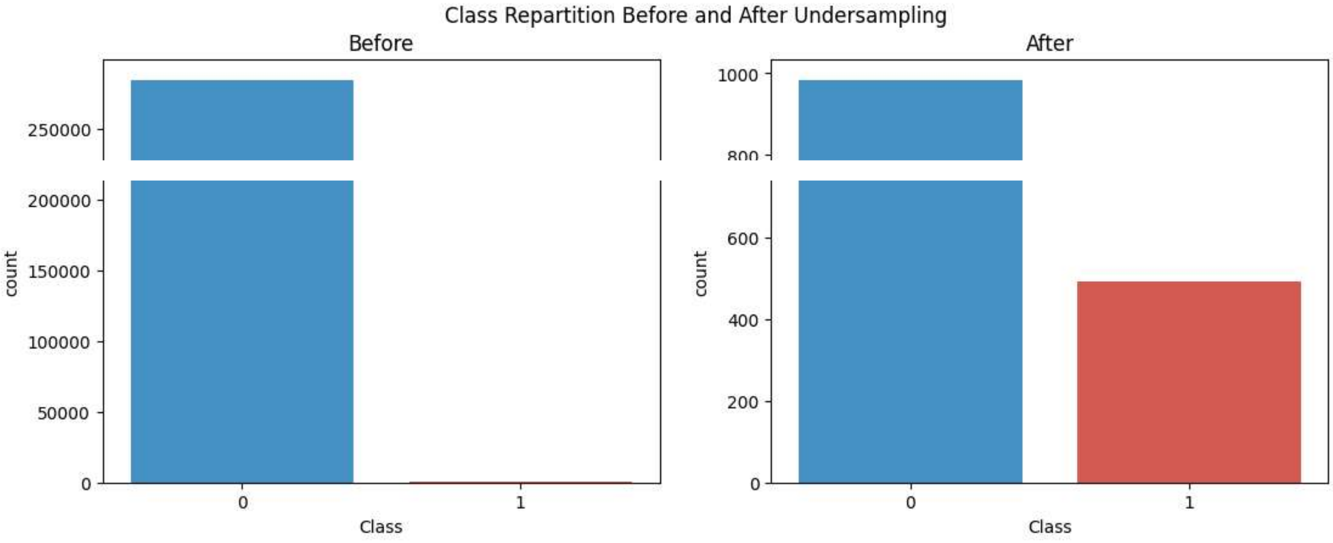
```
<Axes: xlabel='Class', ylabel='count'>
```



```
import imblearn
from imblearn.under_sampling import RandomUnderSampler
undersample = RandomUnderSampler(sampling_strategy=0.5)


cols = credit_card_data.columns.tolist()
cols = [c for c in cols if c not in ["Class"]]
target = "Class"


#define X and Y
X = credit_card_data[cols]
Y = credit_card_data[target]
#undersample
X_under, Y_under = undersample.fit_resample(X, Y)


from pandas import DataFrame
test = pd.DataFrame(Y_under, columns = ['Class'])


import matplotlib.pyplot as plt
import seaborn as sns

fig, axs = plt.subplots(ncols=2, figsize=(13, 4.5))

# Define custom colors for the classes
```

```
class_colors = ['#3498db', '#e74c3c']  # Blue for class 0, Red for class 1

# Plotting before undersampling with custom colors
sns.countplot(x="Class", data=credit_card_data, ax=axs[0], hue="Class", palette=class_colors

# Plotting after undersampling with custom colors
sns.countplot(x="Class", data=test, ax=axs[1], hue="Class", palette=class_colors, legend=Fal

# Set the overall title
fig.suptitle("Class Repartition Before and After Undersampling")

# Set individual titles for subplots
a1 = fig.axes[0]
a1.set_title("Before")

a2 = fig.axes[1]
a2.set_title("After")
```

```
Text(0.5, 1.0, 'After')
```



Train Test Split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_under, Y_under, test_size=0.2, random_
```

Support Vector Machine

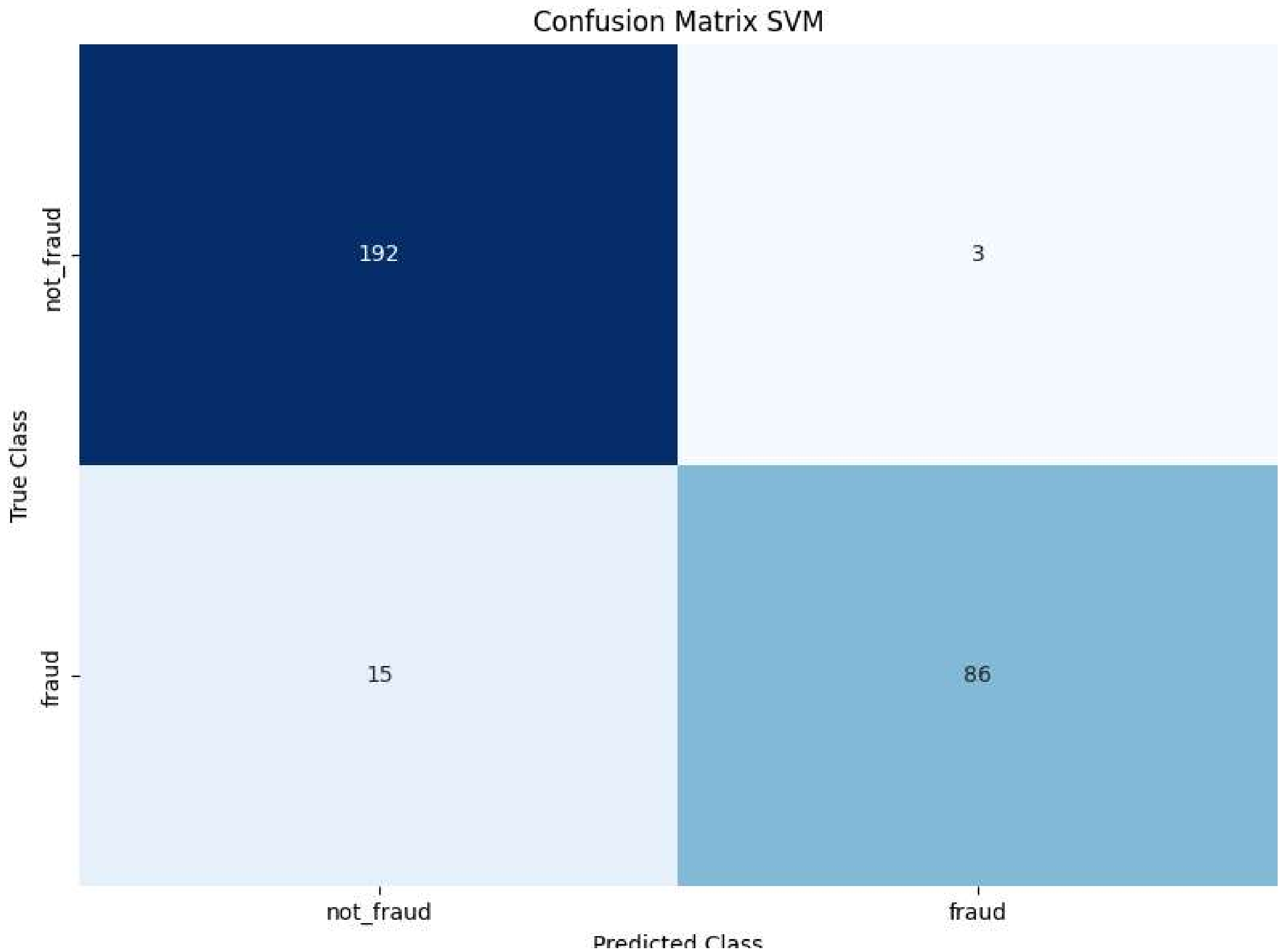
```
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve


model = SVC()


model.fit(X_train,y_train)
```

```
▾ SVC
SVC()
```

```
#train the model
model2 = SVC(probability=True, random_state=2)
svm = model2.fit(X_train, y_train)


#predictions
y_pred_svm = model2.predict(X_test)


#scores
print("Accuracy SVM:",metrics.accuracy_score(y_test, y_pred_svm))
print("Precision SVM:",metrics.precision_score(y_test, y_pred_svm))
print("Recall SVM:",metrics.recall_score(y_test, y_pred_svm))
print("F1 Score SVM:",metrics.f1_score(y_test, y_pred_svm))
```

```
Accuracy SVM: 0.9391891891891891
Precision SVM: 0.9662921348314607
Recall SVM: 0.8514851485148515
F1 Score SVM: 0.9052631578947369
```

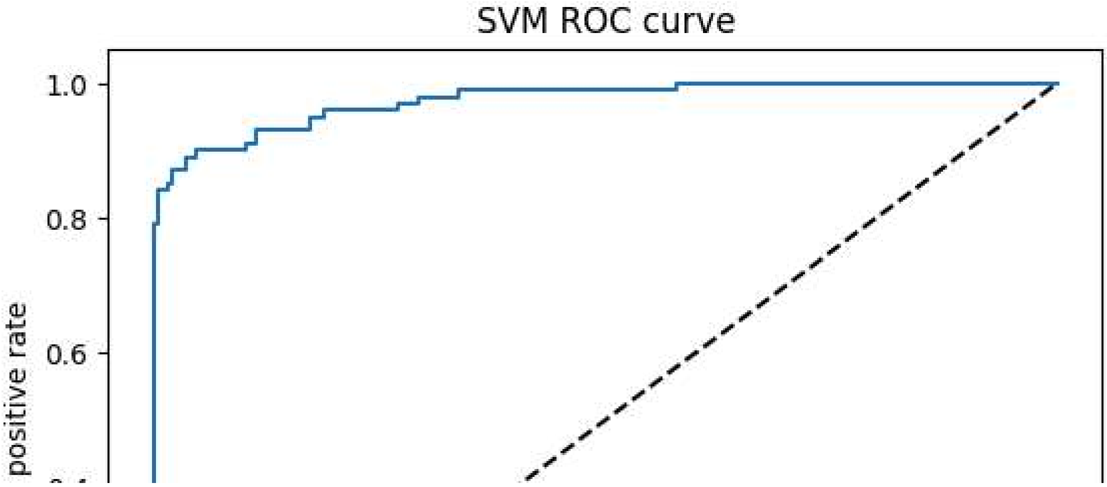
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.metrics import confusion_matrix

# Assuming y_pred_svm contains your model's predictions
matrix_svm = confusion_matrix(y_test, y_pred_svm)

# Create a DataFrame for better heatmap visualization
cm_svm = pd.DataFrame(matrix_svm, index=['not_fraud', 'fraud'], columns=['not_frauc

# Plotting the confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(cm_svm, annot=True, cbar=None, cmap="Blues", fmt='g')

# Adding titles and labels
plt.title("Confusion Matrix SVM")
plt.tight_layout()
plt.ylabel("True Class")
plt.xlabel("Predicted Class")

plt.show()
```



```
#AUC
y_pred_svm_proba = model2.predict_proba(X_test)[::,1]
fpr_svm, tpr_svm, _ = metrics.roc_curve(y_test, y_pred_svm_proba)
auc_svm = metrics.roc_auc_score(y_test, y_pred_svm_proba)
print("AUC SVM :", auc_svm)
```

```
AUC SVM : 0.9747651688245748
```

```
#ROC
plt.plot(fpr_svm,tpr_svm,label="SVM, auc={:.3f})".format(auc_svm))
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('SVM ROC curve')
plt.legend(loc=4)
plt.show()
```


SVM ROC curve
1.0
0.8
0.6
positive rate