Name and USN:

- Swaraj Khan (ENG21CS0433)
- V Kartik (ENG21CS0456)

Dataset

- Credit Card Fraud Detection

Team number:

- 001

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv('creditcard.csv')
df.head()
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2

5 rows × 31 columns

In [ ]:
```
df.describe()
```

Out[ ]:

	Time	V1	V2	V3	V4	
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.8
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.6
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.3
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.1
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.9
50%	84692.000000	1.810880e-02	6.548556e-02	1.798463e-01	-1.984653e-02	-5.4
75%	139320.500000	1.315642e+00	8.037239e-01	1.027196e+00	7.433413e-01	6.1
max	172792.000000	2.454930e+00	2.205773e+01	9.382558e+00	1.687534e+01	3.4

8 rows × 31 columns

In [ ]:

```
print(df.shape)
print(df.size)
```

```
(284807, 31)
8829017
```

In [ ]:

```
missing_values = df.isnull().sum()
print(missing_values)
```

```
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
```

In [ ]:

```
summary_stats = df.describe()
print(summary_stats)
```

```
                Time            V1            V2            V3            V4  \
count  284807.000000  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05   
mean    94813.859575  1.168375e-15  3.416908e-16 -1.379537e-15  2.074095e-15   
std     47488.145955  1.958696e+00  1.651309e+00  1.516255e+00  1.415869e+00   
min         0.000000 -5.640751e+01 -7.271573e+01 -4.832559e+01 -5.683171e+00   
25%     54201.500000 -9.203734e-01 -5.985499e-01 -8.903648e-01 -8.486401e-01   
50%     84692.000000  1.810880e-02  6.548556e-02  1.798463e-01 -1.984653e-02   
75%    139320.500000  1.315642e+00  8.037239e-01  1.027196e+00  7.433413e-01   
max    172792.000000  2.454930e+00  2.205773e+01  9.382558e+00  1.687534e+01   

                 V5            V6            V7            V8            V9  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05   
mean   9.604066e-16  1.487313e-15 -5.556467e-16  1.213481e-16 -2.406331e-15   
std    1.380247e+00  1.332271e+00  1.237094e+00  1.194353e+00  1.098632e+00   
min   -1.137433e+02 -2.616051e+01 -4.355724e+01 -7.321672e+01 -1.343407e+01   
25%   -6.915971e-01 -7.682956e-01 -5.540759e-01 -2.086297e-01 -6.430976e-01   
50%   -5.433583e-02 -2.741871e-01  4.010308e-02  2.235804e-02 -5.142873e-02   
75%    6.119264e-01  3.985649e-01  5.704361e-01  3.273459e-01  5.971390e-01   
max    3.480167e+01  7.330163e+01  1.205895e+02  2.000721e+01  1.559499e+01   

       ...           V21           V22           V23           V24  \
count  ...  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05   
mean   ...  1.654067e-16 -3.568593e-16  2.578648e-16  4.473266e-15   
std    ...  7.345240e-01  7.257016e-01  6.244603e-01  6.056471e-01   
min    ... -3.483038e+01 -1.093314e+01 -4.480774e+01 -2.836627e+00   
25%    ... -2.283949e-01 -5.423504e-01 -1.618463e-01 -3.545861e-01   
50%    ... -2.945017e-02  6.781943e-03 -1.119293e-02  4.097606e-02   
75%    ...  1.863772e-01  5.285536e-01  1.476421e-01  4.395266e-01   
max    ...  2.720284e+01  1.050309e+01  2.252841e+01  4.584549e+00   

                V25           V26           V27           V28         Amount  \
count  2.848070e+05  2.848070e+05  2.848070e+05  2.848070e+05  284807.000000   
mean   5.340915e-16  1.683437e-15 -3.660091e-16 -1.227390e-16      88.349619   
std    5.212781e-01  4.822270e-01  4.036325e-01  3.300833e-01     250.120109   
min   -1.029540e+01 -2.604551e+00 -2.256568e+01 -1.543008e+01       0.000000   
25%   -3.171451e-01 -3.269839e-01 -7.083953e-02 -5.295979e-02       5.600000   
50%    1.659350e-02 -5.213911e-02  1.342146e-03  1.124383e-02      22.000000   
75%    3.507156e-01  2.409522e-01  9.104512e-02  7.827995e-02      77.165000   
max    7.519589e+00  3.517346e+00  3.161220e+01  3.384781e+01   25691.160000   

               Class  
count  284807.000000  
mean        0.001727  
std         0.041527  
min         0.000000  
25%         0.000000  
50%         0.000000  
75%         0.000000  
max         1.000000  

[8 rows x 31 columns]
```

In [ ]:

```
fraud = df[df['Class'] == 1].describe().T
nofraud = df[df['Class'] == 0].describe().T

colors = ['#FFD700','#3B3B3C']

fig,ax = plt.subplots(nrows = 2,ncols = 2,figsize = (5,15))
plt.subplot(2,2,1)
sns.heatmap(fraud[['mean']][:15],annot = True,cmap = colors,linewidths = 0.5,lin
```

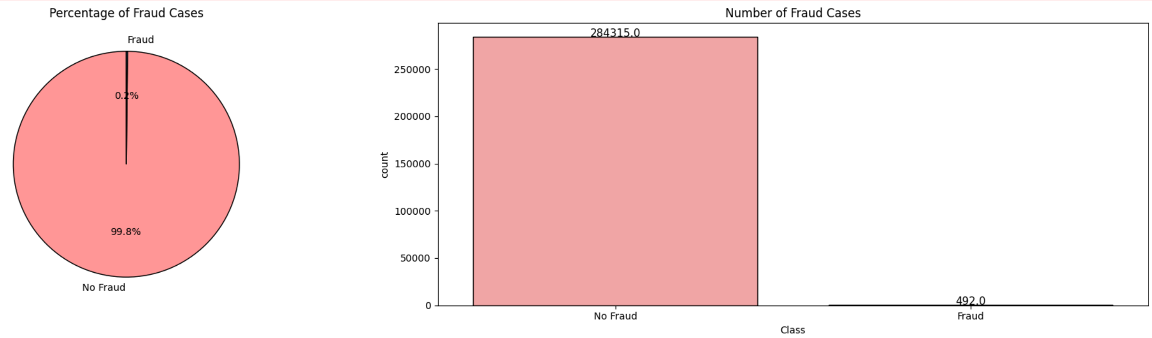
```
plt.title('Fraud Samples : Part 1');

plt.subplot(2,2,2)
sns.heatmap(fraud[['mean']][15:30],annot = True,cmap = colors,linewidths = 0.5,l
plt.title('Fraud Samples : Part 2');

plt.subplot(2,2,3)
sns.heatmap(nofraud[['mean']][:15],annot = True,cmap = colors,linewidths = 0.5,l
plt.title('No Fraud Samples : Part 1');

plt.subplot(2,2,4)
sns.heatmap(nofraud[['mean']][15:30],annot = True,cmap = colors,linewidths = 0.5
plt.title('No Fraud Samples : Part 2');

fig.tight_layout(w_pad = 2)
```


Fraud Samples : Part 1
Time 80746.81
V1 -4.77
V2 3.62
V3 -7.03
V4 4.54
V5 -3.15
V6 -1.40
V7 -5.57
V8 0.57
V9 -2.58
V10 -5.68
V11 3.80
V12 -6.26
V13 -0.11
V14 -6.97
mean
Fraud Samples : Part 2
V15 -0.09
V16 -4.14
V17 -6.67
V18 -2.25
V19 0.68
V20 0.37
V21 0.71
V22 0.01
V23 -0.04
V24 -0.11
V25 0.04
V26 0.05
V27 0.17
V28 0.08
Amount 122.21
mean
No Fraud Samples : Part 1
Time 94838.20
V1 0.01
V2 -0.01
V3 0.01
V4 -0.01
No Fraud Samples : Part 2
V15 0.00
V16 0.01
V17 0.01
V18 0.00
V19 -0.00

In [ ]:

```
import matplotlib.pyplot as plt
import seaborn as sns

fraud_percentage = df['Class'].value_counts(normalize=True) * 100

colors = ['#ff9999', '#66b3ff']

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=(20, 5))

plt.subplot(1, 2, 1)
plt.pie(fraud_percentage, labels=['No Fraud', 'Fraud'], autopct='%1.1f%%', start
        wedgeprops={'edgecolor': 'black', 'linewidth': 1, 'antialiased': True})
plt.title('Percentage of Fraud Cases')

plt.subplot(1, 2, 2)
ax = sns.countplot(x='Class', data=df, edgecolor='black', palette=colors)
for rect in ax.patches:
    ax.text(rect.get_x() + rect.get_width() / 2, rect.get_height() + 2, rect.get
            horizontalalignment='center', fontsize=11)
ax.set_xticklabels(['No Fraud', 'Fraud'])
plt.title('Number of Fraud Cases')

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

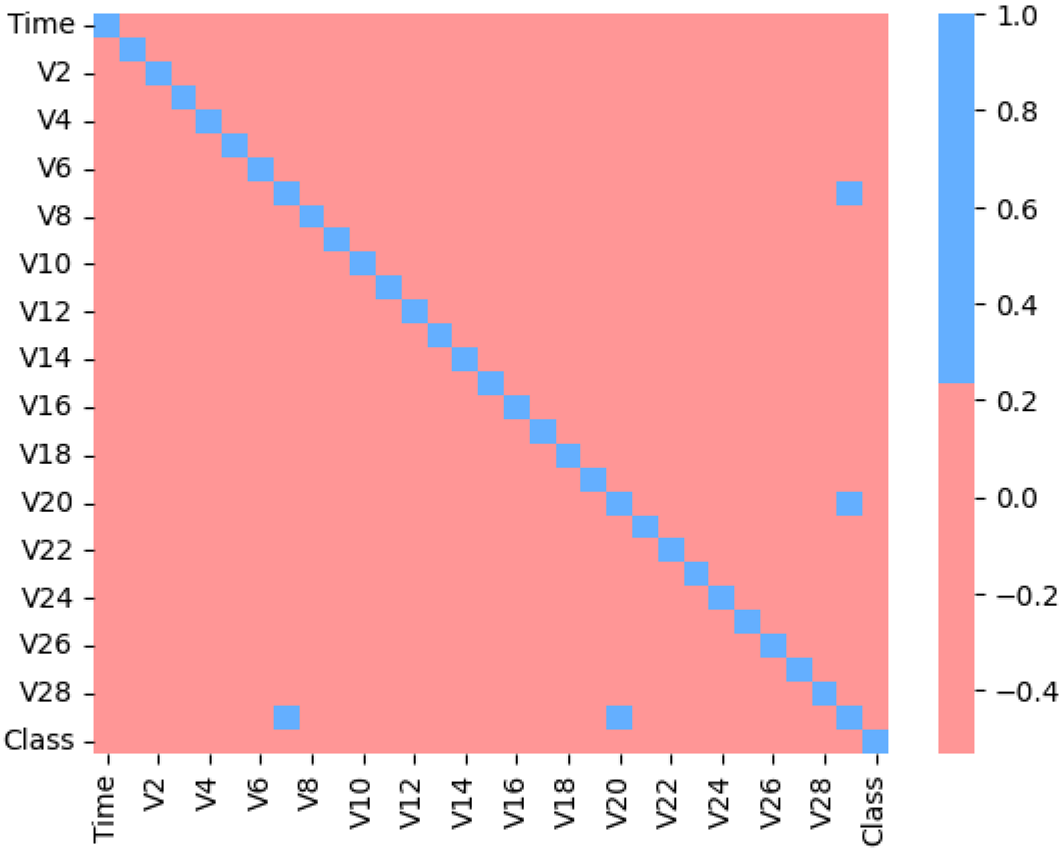
```
C:\Users\Swaraj\AppData\Local\Temp\ipykernel_11532\3349867560.py:21: FutureWarnin
g:

Passing `palette` without assigning `hue` is deprecated and will be removed in v
0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effe
ct.

  ax = sns.countplot(x='Class', data=df, edgecolor='black', palette=colors)  # Ex
plicitly specify 'x' parameter
C:\Users\Swaraj\AppData\Local\Temp\ipykernel_11532\3349867560.py:25: UserWarning:
set_ticklabels() should only be used with a fixed number of ticks, i.e. after set
_ticks() or using a FixedLocator.
  ax.set_xticklabels(['No Fraud', 'Fraud'])
```



In [ ]:

```
sns.heatmap(data.corr(),cmap = colors,cbar = True)
```

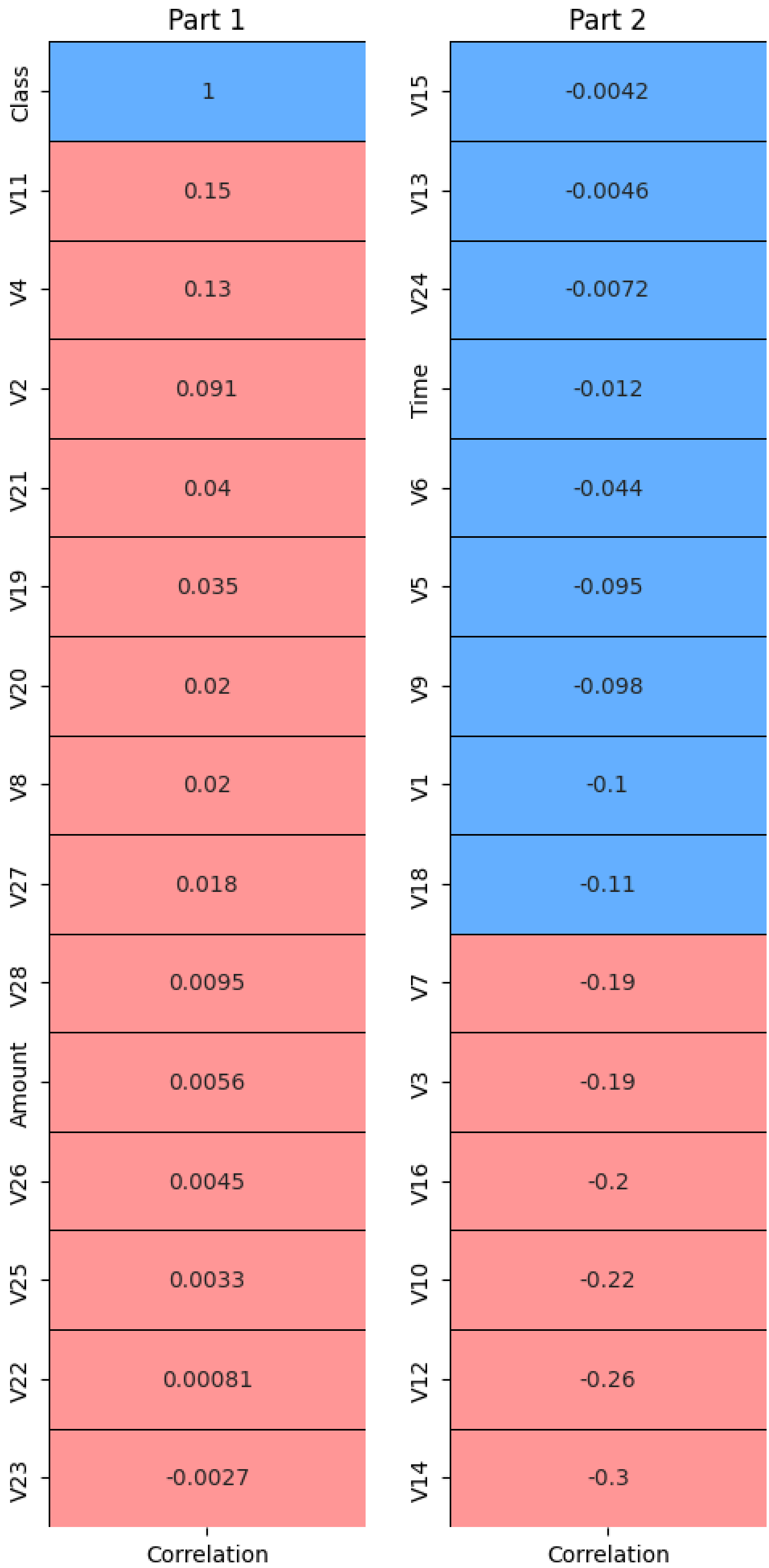
Out[ ]: `<Axes: >`



In [ ]:

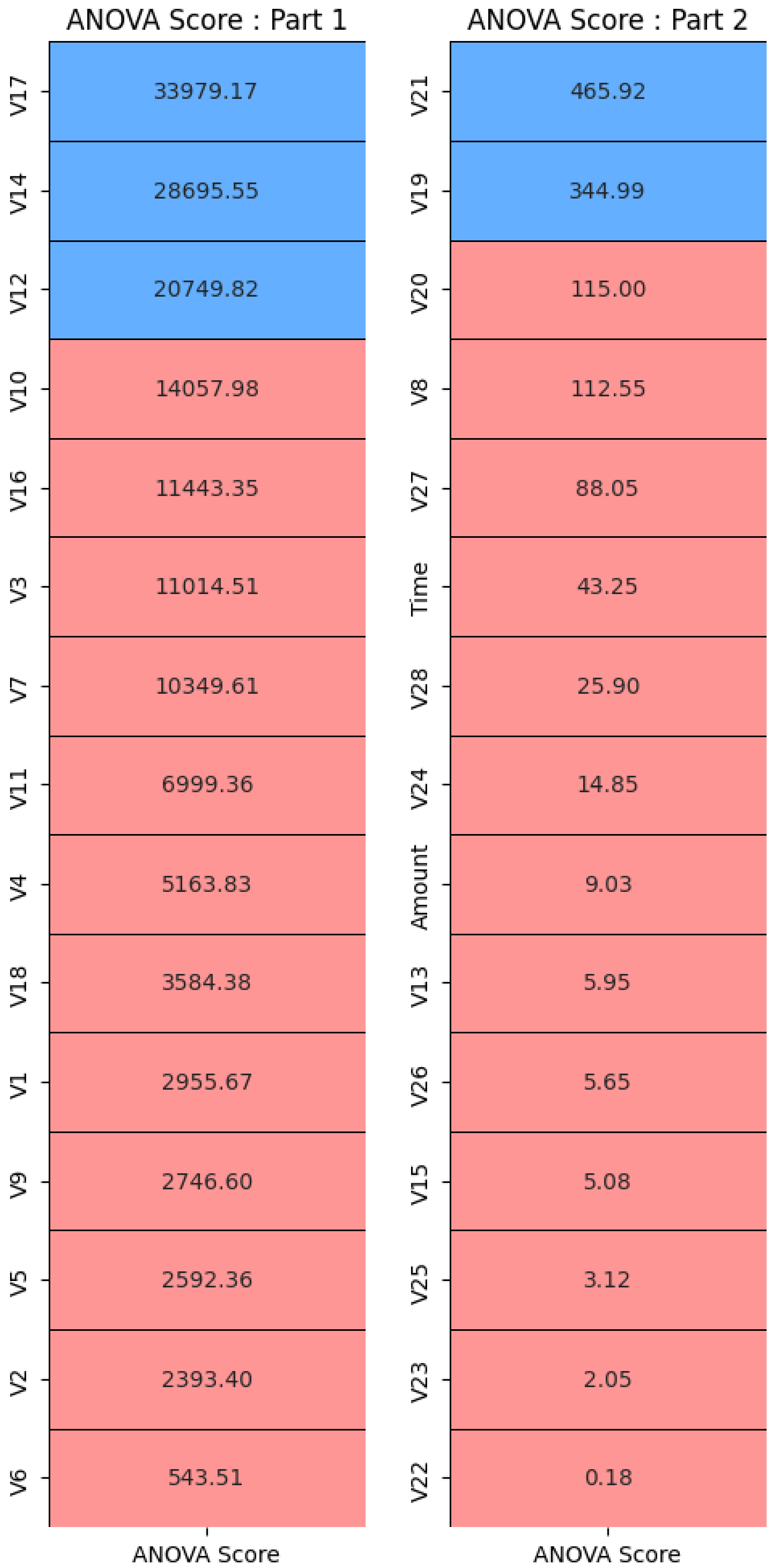
```
corr = data.corrwith(data['Class']).sort_values(ascending = False).to_frame()
corr.columns = ['Correlation']
fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (5,10))

plt.subplot(1,2,1)
```

```
sns.heatmap(corr.iloc[:15,:],annot = True,cmap = colors,linewidths = 0.4,linecol
plt.title('Part 1')

plt.subplot(1,2,2)
sns.heatmap(corr.iloc[15:30],annot = True,cmap = colors,linewidths = 0.4,linecol
plt.title('Part 2')

fig.tight_layout(w_pad = 2)
```


Part 1
Part 2
Class 1
V11 0.15
V4 0.13
V2 0.091
V21 0.04
V19 0.035
V20 0.02
V8 0.02
V27 0.018
V28 0.0095
Amount 0.0056
V26 0.0045
V25 0.0033
V22 0.00081
V23 -0.0027
V15 -0.0042
V13 -0.0046
V24 -0.0072
Time -0.012
V6 -0.044
V5 -0.095
V9 -0.098
V1 -0.1
V18 -0.11
V7 -0.19
V3 -0.19
V16 -0.2
V10 -0.22
V12 -0.26
V14 -0.3
Correlation
Correlation

In [ ]:
```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import f_classif
```

In [ ]:
```
features = data.loc[:,:'Amount']
target = data.loc[:,'Class']

best_features = SelectKBest(score_func = f_classif,k = 'all')
fit = best_features.fit(features,target)

featureScores = pd.DataFrame(data = fit.scores_,index = list(features.columns),c
featureScores = featureScores.sort_values(ascending = False,by = 'ANOVA Score')

fig,ax = plt.subplots(nrows = 1,ncols = 2,figsize = (5,10))

plt.subplot(1,2,1)
sns.heatmap(featureScores.iloc[:15,:],annot = True,cmap = colors,linewidths = 0.
plt.title('ANOVA Score : Part 1')

plt.subplot(1,2,2)
sns.heatmap(featureScores.iloc[15:30],annot = True,cmap = colors,linewidths = 0.
plt.title('ANOVA Score : Part 2')

fig.tight_layout(w_pad = 2)
```


ANOVA Score : Part 1
V17 33979.17
V14 28695.55
V12 20749.82
V10 14057.98
V16 11443.35
V3 11014.51
V7 10349.61
V11 6999.36
V4 5163.83
V18 3584.38
V1 2955.67
V9 2746.60
V5 2592.36
V2 2393.40
V6 543.51
ANOVA Score
ANOVA Score : Part 2
V21 465.92
V19 344.99
V20 115.00
V8 112.55
V27 88.05
Time 43.25
V28 25.90
V24 14.85
Amount 9.03
V13 5.95
V26 5.65
V15 5.08
V25 3.12
V23 2.05
V22 0.18
ANOVA Score

In [ ]:
```
df1 = data[['V3','V4','V7','V10','V11','V12','V14','V16','V17','Class']].copy(de
df1.head()
```

Out[ ]:

	V3	V4	V7	V10	V11	V12	V14	V16
0	2.536347	1.378155	0.239599	0.090794	-0.551600	-0.617801	-0.311169	-0.470401
1	0.166480	0.448154	-0.078803	-0.166974	1.612727	1.065235	-0.143772	0.463917
2	1.773209	0.379780	0.791461	0.207643	0.624501	0.066084	-0.165946	-2.890083
3	1.792993	-0.863291	0.237609	-0.054952	-0.226487	0.178228	-0.287924	-1.059647
4	1.548718	0.403034	0.592941	0.753074	-0.822843	0.538196	-1.119670	-0.451449

In [ ]:
```
df2 = data.copy(deep = True)
df2.drop(columns = list(featureScores.index[20:]),inplace = True)
df2.head()
```

Out[ ]:

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

5 rows × 21 columns

In [ ]:
```
import imblearn
from collections import Counter
from imblearn.over_sampling import SMOTE
from imblearn.under_sampling import RandomUnderSampler
from imblearn.pipeline import Pipeline
```

In [ ]:
```
over = SMOTE(sampling_strategy = 0.5)
under = RandomUnderSampler(sampling_strategy = 0.1)
f1 = df1.iloc[:,:9].values
t1 = df1.iloc[:,9].values

steps = [('under', under),('over', over)]
pipeline = Pipeline(steps=steps)
f1, t1 = pipeline.fit_resample(f1, t1)
Counter(t1)
```

Out[ ]:
```
Counter({0: 4920, 1: 2460})
```

In [ ]:
```
over = SMOTE(sampling_strategy = 0.5)
under = RandomUnderSampler(sampling_strategy = 0.1)
f2 = df2.iloc[:,:20].values
t2 = df2.iloc[:,20].values

steps = [('under', under),('over', over)]
```

```
pipeline = Pipeline(steps=steps)
f2, t2 = pipeline.fit_resample(f2, t2)
Counter(t2)
```

Out[ ]: Counter({0: 4920, 1: 2460})

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score
from sklearn.metrics import RocCurveDisplay
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.metrics import precision_recall_curve
```

In [ ]:
```
x_train1, x_test1, y_train1, y_test1 = train_test_split(f1, t1, test_size = 0.20
x_train2, x_test2, y_train2, y_test2 = train_test_split(f2, t2, test_size = 0.20
```

In [ ]:
```
def model(classifier,x_train,y_train,x_test,y_test):

    classifier.fit(x_train,y_train)
    prediction = classifier.predict(x_test)
    cv = RepeatedStratifiedKFold(n_splits = 10,n_repeats = 3,random_state = 1)
    print("Cross Validation Score : ",'{0:.2%}'.format(cross_val_score(classifie
    print("ROC_AUC Score : ",'{0:.2%}'.format(roc_auc_score(y_test,prediction)))
    plot_roc_curve(classifier, x_test,y_test)
    plt.title('ROC_AUC_Plot')
    plt.show()

def model_evaluation(classifier,x_test,y_test):

    cm = confusion_matrix(y_test,classifier.predict(x_test))
    names = ['True Neg','False Pos','False Neg','True Pos']
    counts = [value for value in cm.flatten()]
    percentages = ['{0:.2%}'.format(value) for value in cm.flatten()/np.sum(cm)]
    labels = [f'{v1}\n{v2}\n{v3}' for v1, v2, v3 in zip(names,counts,percentages
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cm,annot = labels,cmap = 'Blues',fmt ='')

    print(classification_report(y_test,classifier.predict(x_test)))
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

classifier_lr = LogisticRegression(random_state=0, C=10, penalty='l2')
classifier_lr.fit(x_train1, y_train1)

predictions = classifier_lr.predict(x_test1)
print("Based on Correlation Plot")
print("Cross Validation Score: ", '{0:.2%}'.format(cross_val_score(classifier_lr

probs = classifier_lr.predict_proba(x_test1)[:, 1]

print("ROC_AUC Score: ", '{0:.2%}'.format(roc_auc_score(y_test1, probs)))
```

```
Based on Correlation Plot
Cross Validation Score:  98.46%
ROC_AUC Score:  98.43%
```

In [ ]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

classifier_lr = LogisticRegression(random_state=0, C=10, penalty='l2')
classifier_lr.fit(x_train2, y_train2)

predictions = classifier_lr.predict(x_test2)
print("Based on ANOVA Score")
print("Cross Validation Score: ", '{0:.2%}'.format(cross_val_score(classifier_lr

probs = classifier_lr.predict_proba(x_test2)[:, 1]

print("ROC_AUC Score: ", '{0:.2%}'.format(roc_auc_score(y_test2, probs)))
```

```
Based on ANOVA Score
Cross Validation Score:  98.58%
ROC_AUC Score:  99.02%
```

In [ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

classifier_svm = SVC(probability=True, random_state=0, C=10, kernel='rbf')
classifier_svm.fit(x_train1, y_train1)

predictions = classifier_svm.predict(x_test1)
print("Based on Correlation Plot")

print("Cross Validation Score: ", '{0:.2%}'.format(cross_val_score(classifier_sv

probs = classifier_svm.predict_proba(x_test1)[:, 1]

print("ROC_AUC Score: ", '{0:.2%}'.format(roc_auc_score(y_test1, probs)))
```

```
Based on Correlation Plot
Cross Validation Score:  98.44%
ROC_AUC Score:  98.71%
```

In [ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import roc_auc_score
from sklearn.model_selection import cross_val_score, StratifiedKFold

cv = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)

classifier_svc = SVC(probability=True, random_state=0, C=10, kernel='rbf')
classifier_svc.fit(x_train2, y_train2)

predictions = classifier_svc.predict(x_test2)
print("Based on ANOVA Score")
print("Cross Validation Score: ", '{0:.2%}'.format(cross_val_score(classifier_sv

probs = classifier_svc.predict_proba(x_test2)[:, 1]
```

```
print("ROC_AUC Score: ", '{0:.2%}'.format(roc_auc_score(y_test2, probs)))
```

```
Based on ANOVA Score
Cross Validation Score:  99.22%
ROC_AUC Score:  99.56%
```

For logistic regression:

Sr. No.	ML Algorithm	Cross Validation Score	ROC AUC Score
1	LogisticRegression	98.46%	98.43%
2	LogisticRegression	98.58%	99.02%

For SVM:

Sr. No.	ML Algorithm	Cross Validation Score	ROC AUC Score
1	SVM	98.44%	98.71%
2	SVM	99.22%	99.56%