In [30]:
```
import pandas as pd
import numpy as np
from plotnine import *
```

In [31]:
```
df=pd.read_csv(R"C:\Users\Shobhit Khaiwal\Downloads\creditcard.csv")
```

In [32]:
```
df.head(10)
```

Out[32]:

	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.
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.
5	2.0	-0.425966	0.960523	1.141109	-0.168252	0.420987	-0.029728	0.476201	0.
6	4.0	1.229658	0.141004	0.045371	1.202613	0.191881	0.272708	-0.005159	0.
7	7.0	-0.644269	1.417964	1.074380	-0.492199	0.948934	0.428118	1.120631	-3.
8	7.0	-0.894286	0.286157	-0.113192	-0.271526	2.669599	3.721818	0.370145	0.
9	9.0	-0.338262	1.119593	1.044367	-0.222187	0.499361	-0.246761	0.651583	0.

10 rows × 31 columns

In [33]:
```
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
```

In [34]:
```
df.isnull().sum()
```

Out[34]:
```
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 [35]:
```
df.describe().T
```

Out[35]:

	count	mean	std	min	25%	50%
Time	284807.0	9.481386e+04	47488.145955	0.000000	54201.500000	84692.000000
V1	284807.0	1.168375e-15	1.958696	-56.407510	-0.920373	0.018109
V2	284807.0	3.416908e-16	1.651309	-72.715728	-0.598550	0.065486
V3	284807.0	-1.379537e-15	1.516255	-48.325589	-0.890365	0.179846
V4	284807.0	2.074095e-15	1.415869	-5.683171	-0.848640	-0.019847
V5	284807.0	9.604066e-16	1.380247	-113.743307	-0.691597	-0.054336
V6	284807.0	1.487313e-15	1.332271	-26.160506	-0.768296	-0.274187
V7	284807.0	-5.556467e-16	1.237094	-43.557242	-0.554076	0.040103
V8	284807.0	1.213481e-16	1.194353	-73.216718	-0.208630	0.022358
V9	284807.0	-2.406331e-15	1.098632	-13.434066	-0.643098	-0.051429
V10	284807.0	2.239053e-15	1.088850	-24.588262	-0.535426	-0.092917
V11	284807.0	1.673327e-15	1.020713	-4.797473	-0.762494	-0.032757
V12	284807.0	-1.247012e-15	0.999201	-18.683715	-0.405571	0.140033
V13	284807.0	8.190001e-16	0.995274	-5.791881	-0.648539	-0.013568
V14	284807.0	1.207294e-15	0.958596	-19.214325	-0.425574	0.050601
V15	284807.0	4.887456e-15	0.915316	-4.498945	-0.582884	0.048072
V16	284807.0	1.437716e-15	0.876253	-14.129855	-0.468037	0.066413
V17	284807.0	-3.772171e-16	0.849337	-25.162799	-0.483748	-0.065676
V18	284807.0	9.564149e-16	0.838176	-9.498746	-0.498850	-0.003636
V19	284807.0	1.039917e-15	0.814041	-7.213527	-0.456299	0.003735
V20	284807.0	6.406204e-16	0.770925	-54.497720	-0.211721	-0.062481
V21	284807.0	1.654067e-16	0.734524	-34.830382	-0.228395	-0.029450
V22	284807.0	-3.568593e-16	0.725702	-10.933144	-0.542350	0.006782
V23	284807.0	2.578648e-16	0.624460	-44.807735	-0.161846	-0.011193
V24	284807.0	4.473266e-15	0.605647	-2.836627	-0.354586	0.040976
V25	284807.0	5.340915e-16	0.521278	-10.295397	-0.317145	0.016594
V26	284807.0	1.683437e-15	0.482227	-2.604551	-0.326984	-0.052139
V27	284807.0	-3.660091e-16	0.403632	-22.565679	-0.070840	0.001342

	count	mean	std	min	25%	50%
V28	284807.0	-1.227390e-16	0.330083	-15.430084	-0.052960	0.011244
Amount	284807.0	8.834962e+01	250.120109	0.000000	5.600000	22.000000
Class	284807.0	1.727486e-03	0.041527	0.000000	0.000000	0.000000

In [36]:
```
print(df.columns.values)
```

```
['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']
```

In [37]:
```
df['Class'].unique()
```

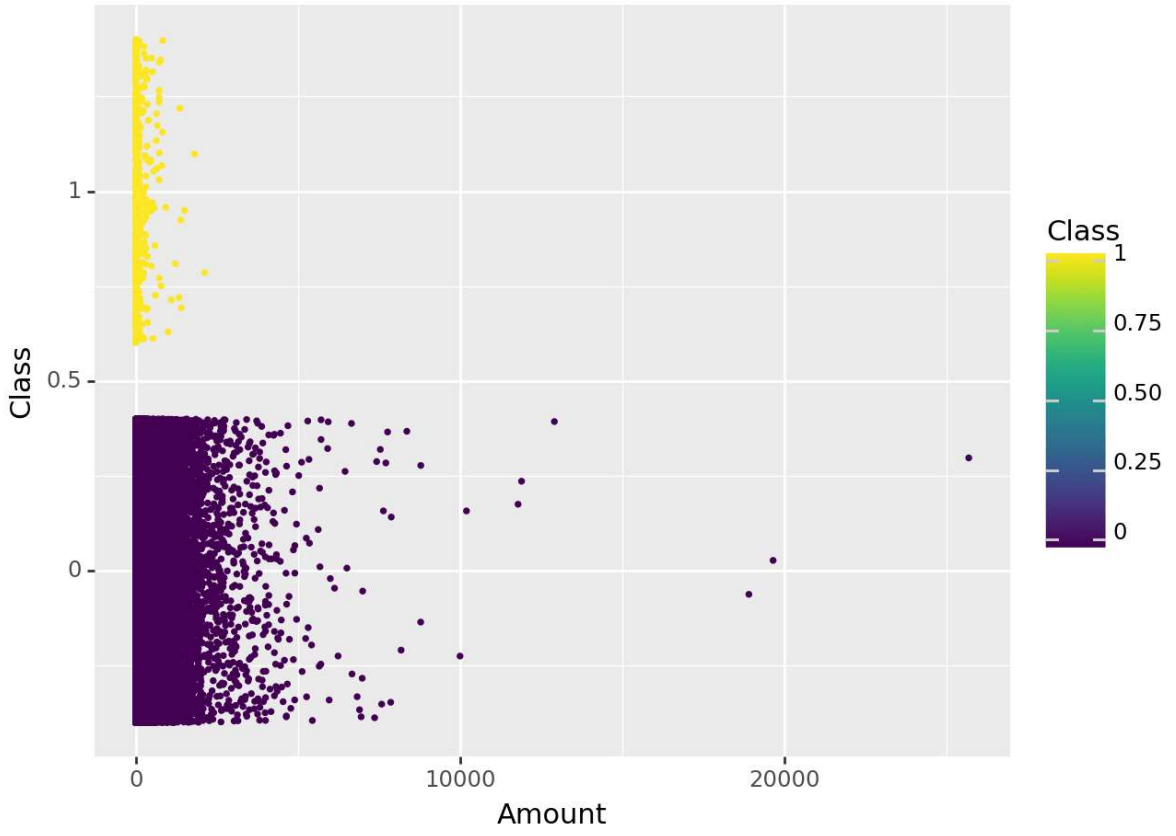
Out[37]:
```
array([0, 1], dtype=int64)
```

In [38]:
```
df['Class'].value_counts()
```

Out[38]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

In [39]:
```
normal=df[df['Class']==0]
fraud=df[df['Class']==1]
```

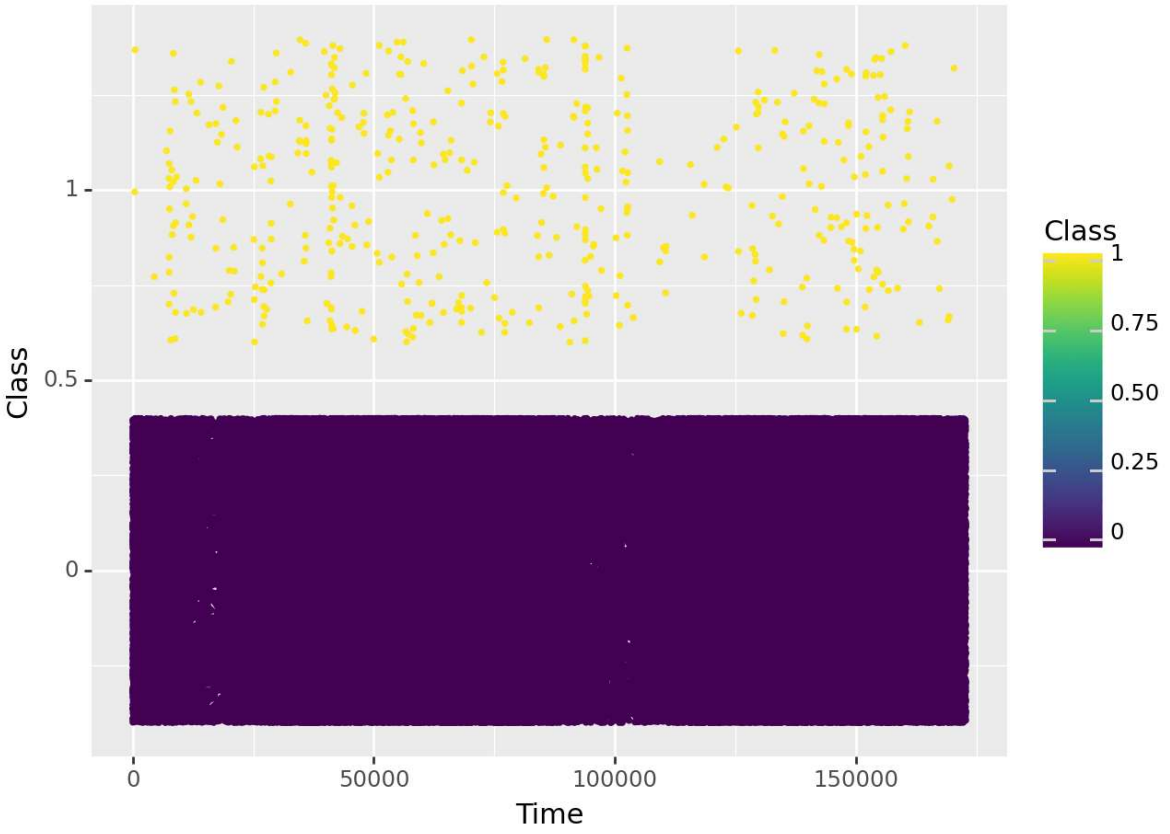
In [40]:
```
(ggplot(df)+aes(x=df['Amount'],y=df['Class'],color='Class')+geom_jitter(size=0.5
```



Out[40]:
```
<Figure Size: (640 x 480)>
```

In [41]:
```
X=pd.DataFrame(columns=['V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11'
'V13','V14','V15','V16','V17','V18','V19','V20','V21','V22','V23','V24',
'V25','V26','V27','V28'])
```

In [42]:
```
(ggplot(df)+aes(x=df['Time'],y=df['Class'],color='Class')+geom_jitter(size=0.5))
```



Out[42]:
```
<Figure Size: (640 x 480)>
```

In [43]:
```
df.drop('Time',axis=1,inplace=True)
```

In [44]:
```
df.duplicated().sum()
```

Out[44]:
```
9144
```

In [45]:
```
d_rows=df.duplicated()
d_data=df[d_rows]
print(d_data)
```

```
              V1        V2        V3        V4        V5        V6        V7  \
33     -0.529912  0.873892  1.347247  0.145457  0.414209  0.100223  0.711206   \
35     -0.535388  0.865268  1.351076  0.147575  0.433680  0.086983  0.693039
113     1.038370  0.127486  0.184456  1.109950  0.441699  0.945283 -0.036715
114     1.038370  0.127486  0.184456  1.109950  0.441699  0.945283 -0.036715
115     1.038370  0.127486  0.184456  1.109950  0.441699  0.945283 -0.036715
...          ...       ...       ...       ...       ...       ...       ...
284708  2.018105  0.073226 -1.615154  0.351058  0.333905 -0.676284  0.050474
284751  2.050734 -0.364010 -2.542843 -0.729357  2.388455  3.318015 -0.479138
284775  1.955547 -0.724606 -1.706511 -0.611145  1.710907  3.914215 -1.248690
284785  0.032887  0.545338 -1.185844 -1.729828  2.932315  3.401529  0.337434
284793  1.971002 -0.699067 -1.697541 -0.617643  1.718797  3.911336 -1.259306

              V8        V9       V10  ...       V21       V22       V23
33      0.176066 -0.286717 -0.484688  ...  0.046949  0.208105 -0.185548  \
35      0.179742 -0.285642 -0.482474  ...  0.049526  0.206537 -0.187108
113     0.350995  0.118950 -0.243289  ...  0.102520  0.605089  0.023092
114     0.350995  0.118950 -0.243289  ...  0.102520  0.605089  0.023092
115     0.350995  0.118950 -0.243289  ...  0.102520  0.605089  0.023092
...          ...       ...       ...  ...       ...       ...       ...
284708 -0.071028  0.302728 -0.193920  ... -0.306646 -0.841768  0.356773
284751  0.791559  0.403374  0.093232  ... -0.293363 -0.834802  0.381740
284775  1.054133  1.314064 -0.150553  ...  0.193605  0.690196  0.155951
284785  0.925377 -0.165663 -0.386953  ... -0.266113 -0.716336  0.108519
284793  1.056209  1.315006 -0.146827  ...  0.188758  0.694418  0.163002

             V24       V25       V26       V27       V28  Amount  Class
33      0.001031  0.098816 -0.552904 -0.073288  0.023307    6.14      0
35      0.000753  0.098117 -0.553471 -0.078306  0.025427    1.77      0
113    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18      0
114    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18      0
115    -0.626463  0.479120 -0.166937  0.081247  0.001192    1.18      0
...          ...       ...       ...       ...       ...     ...    ...
284708  0.655112 -0.331463  0.144352 -0.069820 -0.039845    1.98      0
284751  0.698179 -0.264798  0.219275 -0.052131 -0.066940    0.89      0
284775  0.726775 -0.061219 -0.192666  0.060347 -0.042323   12.99      0
284785  0.688519 -0.460220  0.161939  0.265368  0.090245    1.79      0
284793  0.726365 -0.058282 -0.191813  0.061858 -0.043716    4.99      0

[9144 rows x 30 columns]
```

In [56]:
```
df=df.drop_duplicates()
```

In [57]:
```
x=df.drop('Class',axis=1)
y=df['Class']
```

In [58]:
```
from sklearn.model_selection import train_test_split as tts
```

In [59]:
```
x_train,x_test,y_train,y_test=tts(x,y,test_size=0.3,random_state=45)
```

In [60]:
```
from sklearn.linear_model import LogisticRegression
```

In [61]:
```
lg=LogisticRegression()
```

In [62]:
```
lg.fit(x_train,y_train)
```

```
C:\Users\Shobhit Khaiwal\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\linear_model\_logistic.py:460: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
```

Out[62]:

```
LogisticRegression()
```

In [63]:
```
y_pred=lg.predict(x_test)
```

In [64]:
```
from sklearn.metrics import accuracy_score as acs
```

In [65]:
```
accuracy_pred_y=acs(y_test,y_pred)
```

In [66]:
```
print('accuracy:',accuracy_pred_y)
```

```
accuracy: 0.9989842682499184
```

In [91]:
```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
```

In [92]:
```
def test_multiple_models(x, y, cv):
    models=[
        ("Naive_bayes", GaussianNB()),
        ("Decision Tree", DecisionTreeClassifier()),
        ("Random Forest", RandomForestClassifier()),
        ("Support Vector Machine", SVC())
    ]
    x_train,x_test,y_train,y_test=tts(x,y,test_size=0.3,random_state=45)
    for name, model in models:
        scores = cross_val_score(model, x_train, y_train, cv=cv)
        model.fit(x_train, y_train)
        y_pred = model.predict(x_test)
        accuracy = accuracy_score(y_test, y_pred)
        print(f"Model: {name}")
        print("Cross-Validation Accuracy:", scores.mean())
        print("Accuracy:", accuracy)
        report = classification_report(y_test, y_pred)
        print("Classification Report:\n", report)
        print("----------------------------------------");
```

In [93]:
```
test_multiple_models(x,y,5)
```

```
Model: Naive_bayes
Cross-Validation Accuracy: 0.9774776654605173
Accuracy: 0.9765535254356159
Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.98      0.99     82557
           1       0.06      0.87      0.11       142

    accuracy                           0.98     82699
   macro avg       0.53      0.92      0.55     82699
weighted avg       1.00      0.98      0.99     82699

----------------------------------------
Model: Decision Tree
Cross-Validation Accuracy: 0.9990049955355416
Accuracy: 0.9991172807410005
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     82557
           1       0.74      0.75      0.75       142

    accuracy                           1.00     82699
   macro avg       0.87      0.88      0.87     82699
weighted avg       1.00      1.00      1.00     82699

----------------------------------------
Model: Random Forest
Cross-Validation Accuracy: 0.9994714041803953
Accuracy: 0.999576778437466
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     82557
           1       0.94      0.80      0.87       142

    accuracy                           1.00     82699
   macro avg       0.97      0.90      0.93     82699
weighted avg       1.00      1.00      1.00     82699

----------------------------------------
Model: Support Vector Machine
Cross-Validation Accuracy: 0.9986577804303696
Accuracy: 0.9986940591784665
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00     82557
           1       0.77      0.34      0.47       142

    accuracy                           1.00     82699
   macro avg       0.89      0.67      0.73     82699
weighted avg       1.00      1.00      1.00     82699

----------------------------------------
```

In [ ]: