Machine Learning Clustering and Dimensionality Reduction

Assignment #1

Rhichard Koh

In [ ]:
```
import pandas as pd
```

Apply PCA to the train set

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

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.0986
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3053
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.2948
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.7084
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6791
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.4146

284807 rows × 31 columns

In [ ]:
```
df.columns
```

Out[ ]:
```
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')
```

In [ ]:
```
X = df.drop(columns=['Class'])
X
```

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	0.0986
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.0851
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.2476
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.3774
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.2705
...	...	...	...	...	...	...	...	...	
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.3053
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.2948
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.7084
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.6791
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.4146

284807 rows × 30 columns

In [ ]:

```
X.columns
```

Out[ ]:

```
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'],
      dtype='object')
```

In [ ]:

```
y = df['Class']
y
```

Out[ ]:

```
0         0
1         0
2         0
3         0
4         0
         ..
284802    0
284803    0
284804    0
284805    0
284806    0
Name: Class, Length: 284807, dtype: int64
```

In [ ]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=
```

In [ ]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
```

Visualize Correlation matrix (heatmap) before and after PCA

Before PCA

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sb

X_train_scaled_df = pd.DataFrame(data=X_train_scaled, columns= X_train.columns)
X_train_scaled_df
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	1.022555	0.997851	-0.229626	-0.207385	0.234215	-0.367791	-0.064022	-0.505889	0.03060
1	0.471283	-0.205221	-0.378220	1.027544	-1.424101	-0.078380	0.126364	0.013567	-0.33755
2	1.153387	0.036558	0.495563	-0.370033	-0.500363	0.777856	-0.268414	0.632710	0.09898
3	-0.023638	-0.273682	0.612684	1.156521	1.957021	0.359664	0.750211	0.680997	-0.06809
4	-0.255590	-2.056777	1.145573	-0.283165	-0.019856	-0.617403	-0.358912	-0.351206	1.10558
...	...	...	...	...	...	...	...	...	
227840	-0.403774	0.598760	0.061075	0.324476	0.327051	-0.214298	-0.159001	-0.133772	0.10062
227841	1.352067	-0.396713	0.087171	-0.753679	-0.875555	1.397573	2.926905	-0.375714	1.14500
227842	-0.315815	-0.075332	0.599620	1.007243	0.344127	0.251006	-0.609354	0.864946	-0.33225
227843	-0.144489	-1.506155	1.421728	-1.664055	-2.682097	1.344205	2.040924	-0.380267	1.86593
227844	-0.387707	0.629238	-0.473540	0.255988	-0.492355	-0.745868	-0.476073	-0.404876	-0.15789

227845 rows × 30 columns

In [ ]:
```
X_train_scaled_df.corr()
```

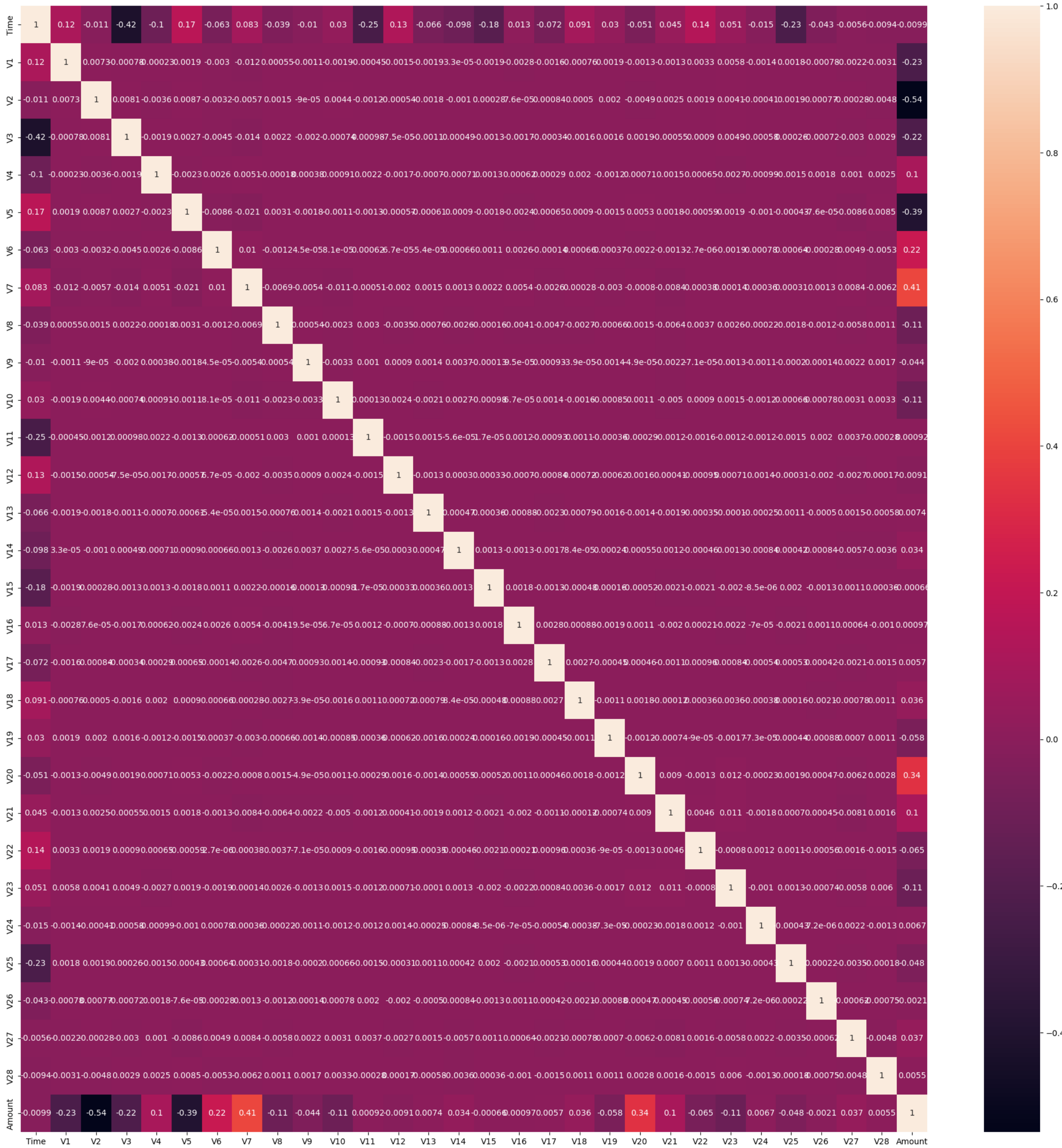
Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	
Time	1.000000	0.117203	-0.010844	-0.421238	-0.104487	0.171204	-0.063108	0.083313	-0.0387
V1	0.117203	1.000000	0.007295	-0.000782	-0.000227	0.001902	-0.003035	-0.011855	0.0005
V2	-0.010844	0.007295	1.000000	0.008112	-0.003559	0.008695	-0.003240	-0.005729	0.0014
V3	-0.421238	-0.000782	0.008112	1.000000	-0.001850	0.002671	-0.004517	-0.014211	0.0022
V4	-0.104487	-0.000227	-0.003559	-0.001850	1.000000	-0.002312	0.002631	0.005144	-0.0001
V5	0.171204	0.001902	0.008695	0.002671	-0.002312	1.000000	-0.008561	-0.020789	0.0031
V6	-0.063108	-0.003035	-0.003240	-0.004517	0.002631	-0.008561	1.000000	0.010363	-0.0011
V7	0.083313	-0.011855	-0.005729	-0.014211	0.005144	-0.020789	0.010363	1.000000	-0.0069
V8	-0.038743	0.000546	0.001479	0.002246	-0.000180	0.003117	-0.001161	-0.006947	1.0000
V9	-0.010310	-0.001120	-0.000090	-0.001977	0.000379	-0.001800	0.000045	-0.005432	0.0005
V10	0.029694	-0.001860	0.004352	-0.000745	0.000909	-0.001075	0.000081	-0.011277	-0.0023
V11	-0.247385	-0.000448	-0.001157	0.000983	0.002178	-0.001301	0.000622	-0.000509	0.0030
V12	0.125590	-0.001497	-0.000544	-0.000075	-0.001699	-0.000569	0.000067	-0.001953	-0.0035
V13	-0.066410	-0.001924	-0.001757	-0.001095	-0.000702	-0.000614	-0.000054	0.001543	-0.0007
V14	-0.098385	0.000033	-0.001041	0.000494	-0.000707	0.000905	0.000656	0.001256	-0.0025
V15	-0.183030	-0.001857	0.000280	-0.001326	0.001253	-0.001777	0.001112	0.002246	-0.0001
V16	0.012980	-0.002791	0.000076	-0.001699	0.000616	-0.002415	0.002639	0.005372	-0.0041
V17	-0.071968	-0.001581	0.000843	-0.000341	0.000288	0.000645	-0.000140	-0.002626	-0.0046
V18	0.091128	-0.000757	0.000502	-0.001566	0.001995	0.000903	0.000657	0.000283	-0.0026
V19	0.029596	0.001869	0.001993	0.001622	-0.001183	-0.001521	0.000371	-0.002998	-0.0006
V20	-0.051246	-0.001306	-0.004863	0.001906	0.000714	0.005301	-0.002250	-0.000800	0.0015
V21	0.045097	-0.001346	0.002492	-0.000554	0.001492	0.001764	-0.001328	-0.008404	-0.0064
V22	0.144891	0.003286	0.001870	0.000905	0.000647	-0.000594	-0.000003	0.000382	0.0037
V23	0.050655	0.005771	0.004074	0.004907	-0.002682	0.001927	-0.001865	0.000138	0.0025
V24	-0.015464	-0.001356	-0.000414	-0.000579	-0.000989	-0.001035	0.000782	0.000365	-0.0002
V25	-0.232573	0.001797	0.001938	0.000258	-0.001542	-0.000433	0.000644	0.000310	-0.0018
V26	-0.042594	-0.000779	0.000773	-0.000720	0.001766	-0.000076	-0.000277	0.001284	-0.0011
V27	-0.005570	-0.002175	-0.000278	-0.003023	0.001041	-0.008595	0.004883	0.008366	-0.0058
V28	-0.009371	-0.003073	-0.004816	0.002885	0.002495	0.008496	-0.005313	-0.006193	0.0011
Amount	-0.009936	-0.233925	-0.536033	-0.218054	0.103808	-0.394113	0.221781	0.411463	-0.1054

30 rows × 30 columns

In [ ]:
```
plt.figure(figsize=(25, 25))

sb.heatmap(X_train_scaled_df.corr(), annot=True)
```

Out[ ]: <Axes: >



After PCA

In [ ]:
```
from sklearn.decomposition import PCA
pca = PCA()
X_train_PCA = pca.fit_transform(X_train_scaled)
X_train_PCA
```

Out[ ]:
```
array([[-0.45171765, -1.29559725,  0.0421551 , ...,  0.03195231,
        -0.13785439,  0.01672698],
       [-0.25134876, -1.31061702, -0.10325889, ..., -0.27767841,
         0.65829119,  0.02265744],
       [-0.40706699, -1.42453283,  0.32045815, ..., -0.28999978,
        -0.18890148,  0.04057583],
       ...,
       [-0.4183145 ,  0.61626946,  0.26434248, ..., -0.6692277 ,
        -0.1581937 ,  0.02980141],
       [-0.58662483, -0.21794326,  0.24625923, ...,  0.69319465,
         0.4484557 , -0.11181253],
       [ 0.12975006,  0.74715442,  0.07854037, ...,  0.04452226,
        -0.20418901, -0.01583862]])
```

In [ ]:
```python
X_train_PCA_df = pd.DataFrame(data=X_train_PCA, columns = X_train.columns)
X_train_PCA_df
```

Out[ ]:

	Time	V1	V2	V3	V4	V5	V6	V7	V
0	-0.451718	-1.295597	0.042155	-0.017005	-0.558668	-0.103815	-0.540228	-0.405253	-1.8870
1	-0.251349	-1.310617	-0.103259	0.402708	-0.246232	0.388712	-1.104457	1.130561	-0.9023
2	-0.407067	-1.424533	0.320458	0.460765	0.406264	0.957740	-0.135362	0.880158	0.3985
3	0.195144	0.444337	0.441017	-0.301536	-0.288001	2.739601	-1.813253	-0.484622	0.2734
4	-0.496166	0.505630	0.756510	0.071279	0.729322	-0.565272	0.492185	-0.453229	-1.0448
...	...	...	...	...	...	...	...	...	...
227840	-0.432766	1.085389	0.103383	0.134239	0.101425	-0.207721	-0.077260	-0.501031	-0.4309
227841	-0.050833	-2.107870	0.347575	1.184980	-0.696713	-0.527504	0.206993	0.797536	-0.6999
227842	-0.418314	0.616269	0.264342	-0.615669	-0.494755	-0.551815	0.203305	1.002262	0.2025
227843	-0.586625	-0.217943	0.246259	0.750738	0.238745	-1.475042	-0.287662	0.415580	-0.0162
227844	0.129750	0.747154	0.078540	-0.175870	0.115263	-0.629163	0.838574	0.687421	0.5429

227845 rows × 30 columns

In [ ]:
```python
X_train_PCA_df.corr()
```

Out[ ]:

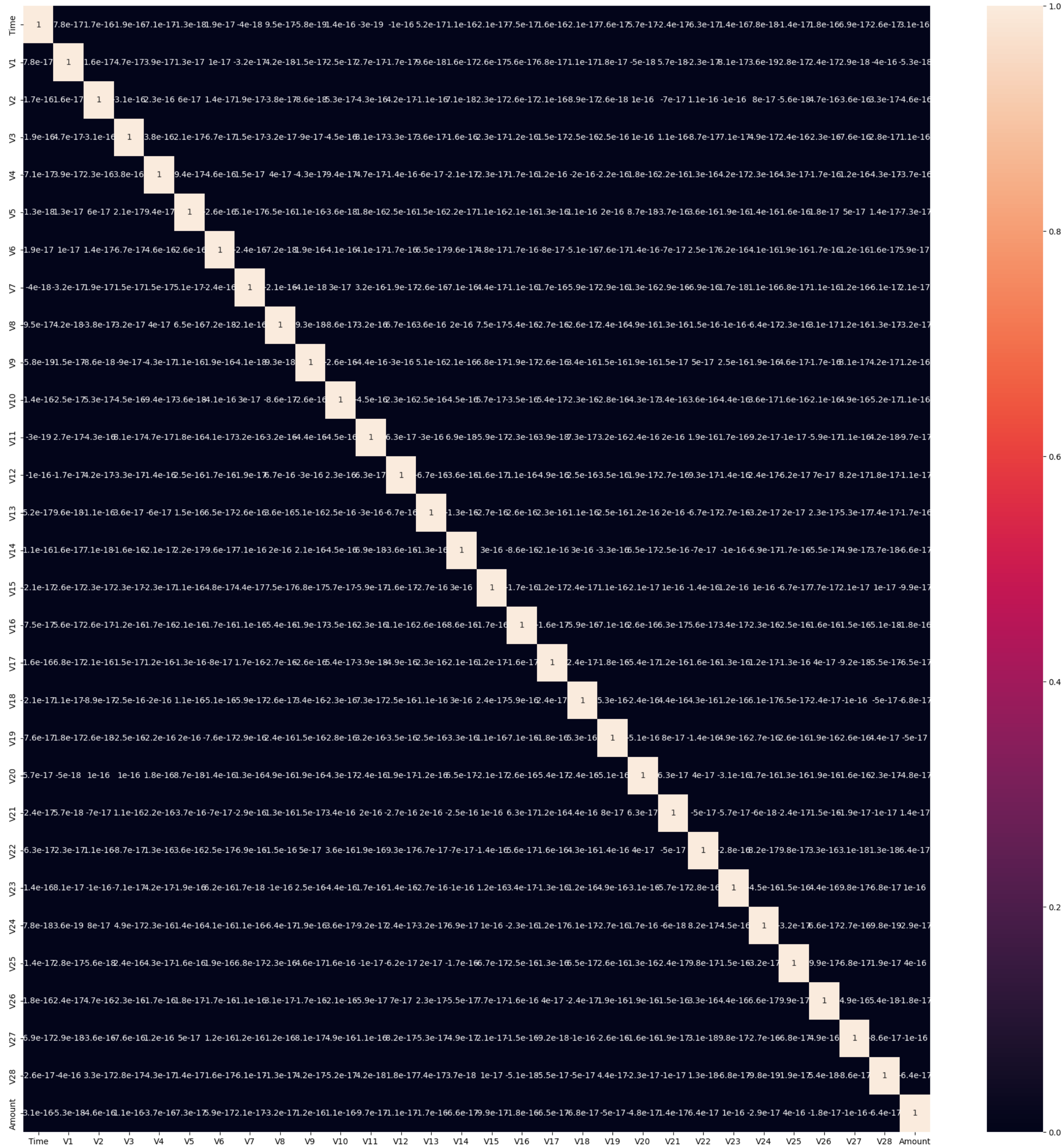
	Time	V1	V2	V3	V4	V5	
Time	1.000000e+00	7.839378e-17	1.708444e-16	-1.868911e-16	-7.130918e-17	-1.278624e-18	1.850
V1	7.839378e-17	1.000000e+00	1.586145e-17	4.744220e-17	3.927787e-17	1.291499e-17	1.004
V2	1.708444e-16	1.586145e-17	1.000000e+00	-3.124562e-16	2.323815e-16	6.032887e-17	1.374
V3	-1.868911e-16	4.744220e-17	-3.124562e-16	1.000000e+00	3.768324e-16	2.144041e-17	-6.7
V4	-7.130918e-17	3.927787e-17	2.323815e-16	3.768324e-16	1.000000e+00	9.352089e-17	-4.6
V5	-1.278624e-18	1.291499e-17	6.032887e-17	2.144041e-17	9.352089e-17	1.000000e+00	-2.5
V6	1.850432e-17	1.004389e-17	1.374008e-17	-6.739788e-17	-4.612447e-16	-2.591059e-16	1.000
V7	-3.954003e-18	-3.221124e-17	1.924877e-17	1.474727e-17	1.499357e-17	5.091816e-17	-2.4
V8	9.460095e-17	4.235341e-18	-3.849014e-17	-3.222983e-17	3.990450e-17	6.472387e-16	-7.2
V9	-5.799076e-19	-1.543606e-17	-8.561077e-18	-8.950723e-17	-4.318201e-17	1.124380e-16	1.890
V10	1.376964e-16	-2.502082e-17	5.300461e-17	-4.520668e-16	-9.448290e-17	-3.576772e-18	-4.1
V11	-3.046093e-19	2.671807e-17	-4.289011e-16	8.111923e-17	4.650798e-17	1.806107e-16	4.113
V12	-1.010646e-16	-1.730079e-17	4.223591e-17	-3.273198e-17	-1.405945e-16	2.476665e-16	-1.6
V13	5.161963e-17	9.557533e-18	-1.139906e-16	3.559297e-17	-5.964028e-17	1.548470e-16	6.481
V14	1.097591e-16	1.582437e-17	7.103984e-18	-1.620744e-16	-2.064736e-17	2.188318e-17	-9.6
V15	-2.124862e-17	2.554684e-17	2.258294e-17	2.283802e-17	-2.277063e-17	1.145358e-16	4.835
V16	-7.528998e-17	5.555474e-17	2.558791e-17	-1.197628e-16	-1.729398e-16	-2.073014e-16	-1.7
V17	1.561661e-16	6.777082e-17	2.066603e-16	1.450569e-17	1.242875e-16	-1.331043e-16	-8.0
V18	-2.146808e-17	1.104800e-17	-8.919885e-17	-2.533323e-16	-1.955507e-16	1.068794e-16	-5.1
V19	-7.599261e-17	1.750763e-17	2.561093e-18	-2.465785e-16	-2.236296e-16	1.954372e-16	-7.5
V20	5.734832e-17	-4.964350e-18	1.024628e-16	1.043208e-16	1.777355e-16	8.674517e-18	-1.4

	Time	V1	V2	V3	V4	V5	
V21	-2.409424e-17	5.719831e-18	-6.970960e-17	1.091305e-16	2.194394e-16	-3.663227e-16	-6.9
V22	-6.295647e-17	-2.254200e-17	1.148360e-16	-8.685002e-17	1.316876e-16	3.648188e-16	2.540
V23	1.446412e-16	8.123133e-17	-1.049816e-16	-7.088991e-17	4.172204e-17	-1.880598e-16	6.161
V24	7.828680e-18	3.572490e-19	8.026503e-17	4.922013e-17	2.317600e-16	1.434056e-16	4.123
V25	-1.392130e-17	2.766950e-17	-5.555925e-18	2.393508e-16	4.345410e-17	-1.597242e-16	1.873
V26	1.799874e-16	2.429678e-17	4.658041e-16	-2.328068e-16	-1.715146e-16	1.783046e-17	-1.7
V27	6.937767e-17	2.880595e-18	-3.584702e-16	-7.615352e-16	1.153877e-16	5.038011e-17	1.191
V28	-2.641047e-17	-4.021459e-16	3.332590e-17	2.806540e-17	-4.258394e-17	1.372546e-17	1.616
Amount	3.141768e-16	-5.310173e-18	-4.573220e-16	1.104138e-16	-3.666818e-16	-7.347193e-17	5.878

30 rows × 30 columns

In [ ]:
```
plt.figure(figsize=(25, 25))

sb.heatmap(X_train_PCA_df.corr(), annot=True)
```

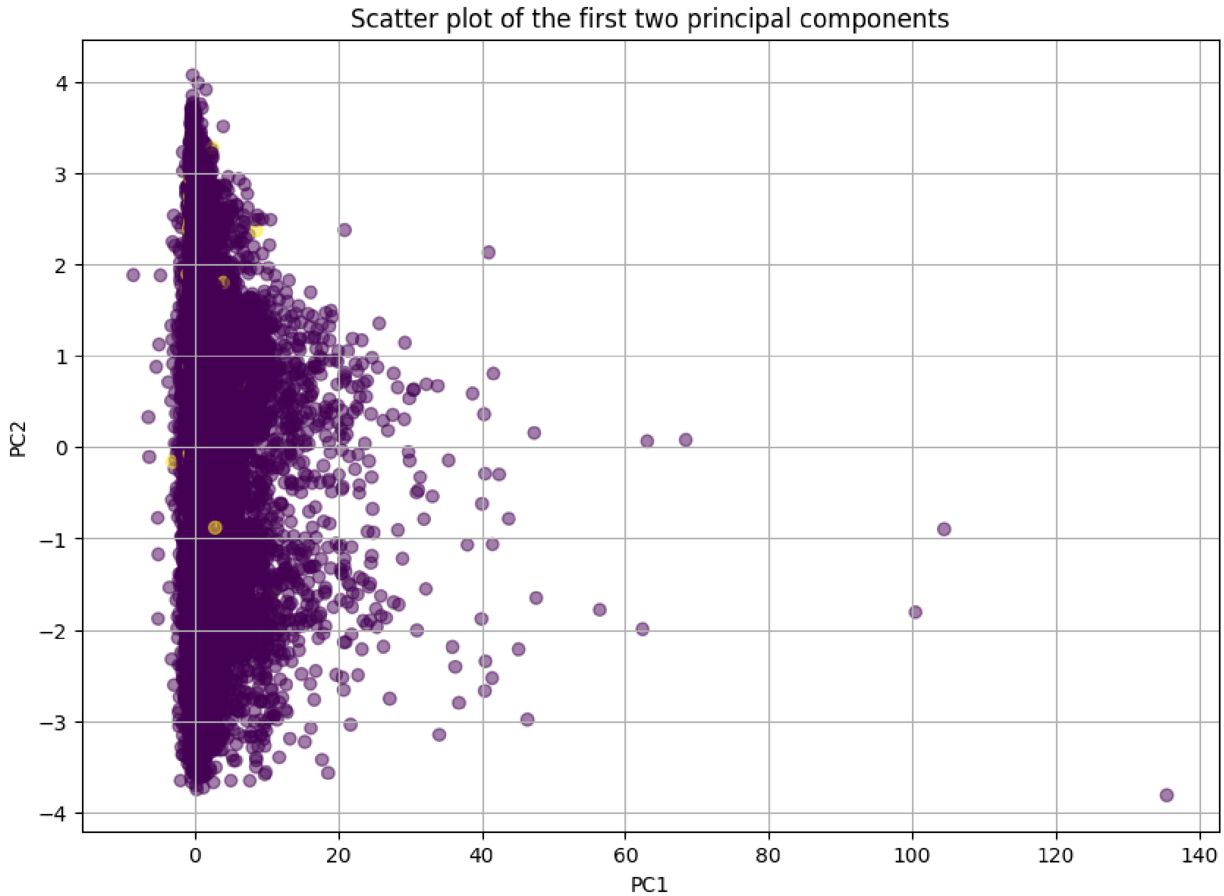
Out[ ]:
```
<Axes: >
```



Visualize the classes in 2D. Discuss corner cases.

In [ ]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 7))

plt.scatter(X_train_PCA_df.iloc[:, 0], X_train_PCA_df.iloc[:, 1], c=y_train, cmap='vir
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('Scatter plot of the first two principal components')
plt.grid(True)
plt.show()
```



Apply SVC on Model 1, raw

In [ ]:
```
from sklearn.svm import SVC
from sklearn.metrics import roc_curve, auc
from sklearn.preprocessing import label_binarize


model = SVC(probability=True)
model.fit(X_train_scaled, y_train)
```

Out[ ]:
```
▾ SVC
SVC(probability=True)
```

In [ ]:
```
X_test_scaled = scaler.transform(X_test)

y_score = model.predict_proba(X_test_scaled)
y_score
```

Out[ ]:
```
array([[2.92213182e-01, 7.07786818e-01],
       [9.99546134e-01, 4.53865621e-04],
       [9.99447428e-01, 5.52572344e-04],
       ...,
       [9.99653336e-01, 3.46663971e-04],
       [9.99671487e-01, 3.28513334e-04],
       [9.99248522e-01, 7.51477857e-04]])
```

In [ ]:
```
y_score_m1 = model.predict_proba(X_test_scaled)[:,1]
fpr_m1, tpr_m1, _ = roc_curve(y_test, y_score_m1)
roc_auc_m1 = auc(fpr_m1, tpr_m1)
```

M2 transformed data. (all features (PCs))

In [ ]:
```
model_2 = SVC(probability=True)
model_2.fit(X_train_PCA, y_train)
```

Out[ ]:
```
▾ SVC
SVC(probability=True)
```

In [ ]:
```
X_test_pca = pca.transform(X_test_scaled)

y_score_m2 = model_2.predict_proba(X_test_pca)[:, 1]
fpr_m2, tpr_m2, _ = roc_curve(y_test, y_score_m2)
roc_auc_m2 = auc(fpr_m2, tpr_m2)
```

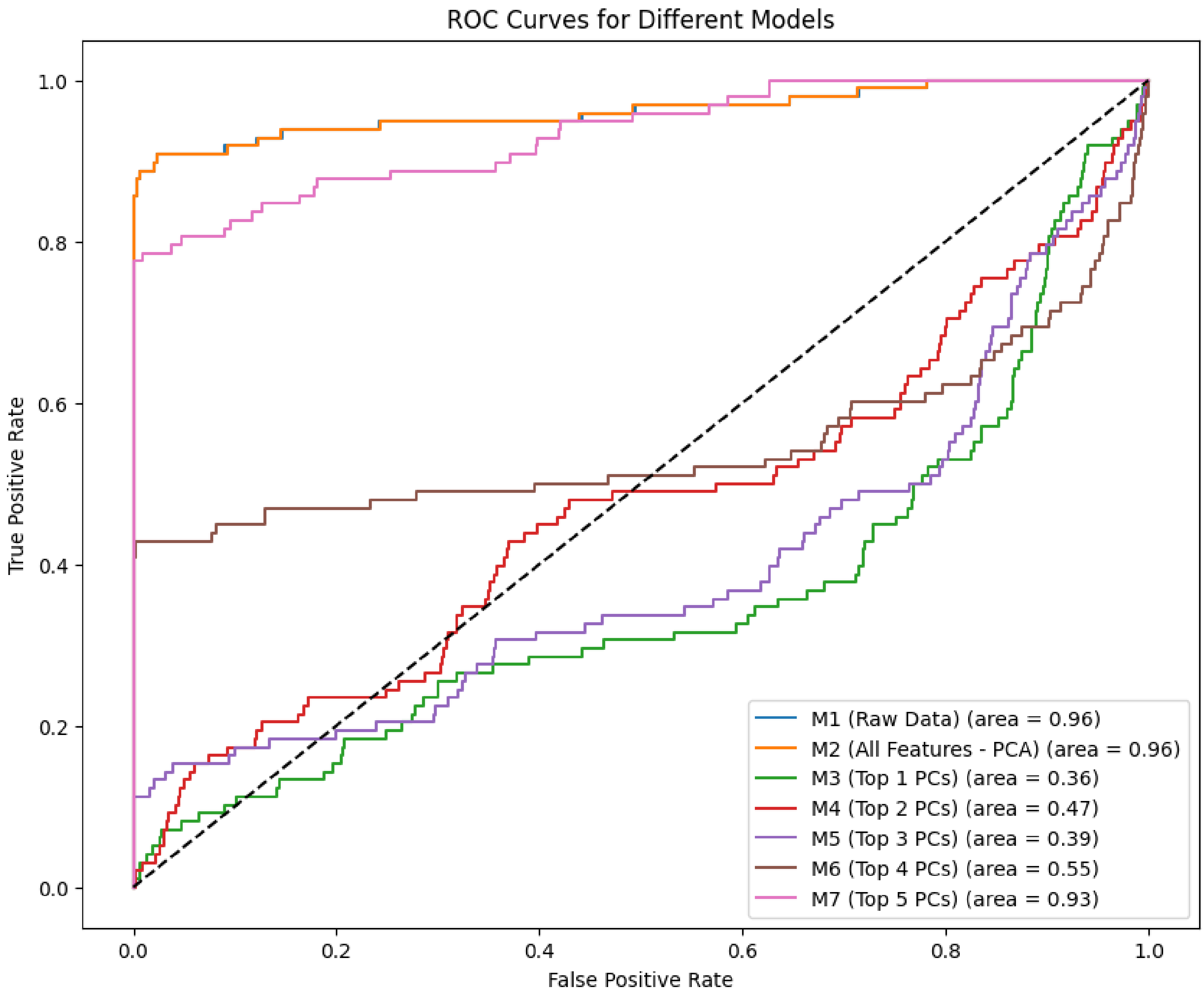
M3-6 reduced data from (top, 1, 2, 3, 4, 5 features (PCs))

In [ ]:
```
models_m3_to_m7 = []

for i in range(1, 6):
    svc = SVC(probability=True)
    svc.fit(X_train_PCA[:, :i], y_train)

    X_test_PCA_reduced = pca.transform(scaler.transform(X_test))[:, :i]

    y_score = svc.predict_proba(X_test_PCA_reduced)[:, 1]

    fpr, tpr, _ = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)

    models_m3_to_m7.append({
        'fpr': fpr,
        'tpr': tpr,
        'roc_auc': roc_auc,
        'label': f'M{i+2} (Top {i} PCs)'
    })
```

Report the findings for all six models in terms of ROC curves on one plot.

In [ ]:
```
plt.figure(figsize=(10, 8))

plt.plot(fpr_m1, tpr_m1, label=f'M1 (Raw Data) (area = {roc_auc_m1:.2f})')

plt.plot(fpr_m2, tpr_m2, label=f'M2 (All Features - PCA) (area = {roc_auc_m2:.2f})')
```

```
for model in models_m3_to_m7:
    plt.plot(model['fpr'], model['tpr'], label=f"{model['label']} (area = {model['roc_

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curves for Different Models')
plt.legend(loc="best")

plt.show()
```



Discuss the merits/de-merits of each model.

Model 1:

Strengths: The model works really well with the untouched data, showing that the original details are quite helpful and a good fit for sorting or categorizing.

Weaknesses: Keeping all the original details could make the model more complicated and might lead to it being too tailored to the training data (overfitting). This could also mean it takes more time to learn from the data and to make predictions, especially with a big dataset.

Model 2:

Strengths: It works as well as Model 1, showing that the PCA method kept most of the important changes in the data. It could also be better at handling random variations and differences in the data than Model 1.

Weaknesses: Like Model 1, using all the components doesn't simplify the model. It might be harder to understand how the model makes decisions because the PCA components don't directly match up with the original details.

Model 3:

Strengths: The model is straightforward and quick to train because it uses only the most important principal component.

Weaknesses: There's a big decrease in how well it performs compared to Models 1 and 2, indicating that relying solely on the top principal component isn't sufficient for precise categorization.

Model 4:

Strengths: It's a bit more complex than Model 3 but remains simpler and quicker than Models 1 and 2.

Weaknesses: The model's AUC (Area Under the Curve) is below 0.5, which means it performs worse than if it were just making random guesses. This suggests that including the top two principal components might add confusing patterns or noise to the model, leading to poor predictions.

Model 5:

Strengths: Using three principal components is more informative than using just one or two, and it also simplifies the model more than using all the original features.

Weaknesses: The model's performance is much worse than random guessing, as indicated by an AUC (Area Under the Curve) significantly lower than 0.5. This suggests that the top three principal components may not be capturing the necessary information for this specific classification task.

Model 6:

Strengths: There's a small improvement compared to Models 4 and 5, which implies that adding the fourth principal component provides some useful information for classifying.

Weaknesses: The model's effectiveness is still close to what you'd expect from random guesses, showing that the top four principal components don't provide enough detail for strong classification results.

Model 7:

Strengths: There's a notable boost in how well it works, nearly matching the levels of Models 1 and 2. This indicates that using the top five principal components gets most of the crucial details needed for sorting or categorizing, and adding just a few more details could further help.

Weaknesses: Although it outperforms Models 3 to 6, it's still not quite as precise as Models 1 and 2. This suggests there could be a balance to find between keeping the model simple and achieving the best possible performance.