[2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

+ Code + Markdown

[3]:

```
df = pd.read_csv('/kaggle/input/creditcardfraud/creditcard.csv')
df.head()
```

[3]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V
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	0.066928	0.128539	-0.189115	0.133558	-0.021
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	-0.339846	0.167170	0.125895	-0.008983	0.014
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	-0.689281	-0.327642	-0.139097	-0.055353	-0.059
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	-1.175575	0.647376	-0.221929	0.062723	0.061
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	0.141267	-0.206010	0.502292	0.219422	0.215

5 rows × 31 columns

[4]:

```
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
```

[5]:

```
df.describe()
```

[5]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V
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	2.848070e+05	...	2.848070e+05	2.848070e+
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	-2.406331e-15	...	1.654067e-16	-3.568593e-
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	1.098632e+00	...	7.345240e-01	7.257016e-
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	-1.343407e+01	...	-3.483038e+01	-1.093314e+
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	-6.430976e-01	...	-2.283949e-01	-5.423504e-
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	-5.142873e-02	...	-2.945017e-02	6.781943e-
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	5.971390e-01	...	1.863772e-01	5.285536e-
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	1.559499e+01	...	2.720284e+01	1.050309e+

8 rows × 31 columns

Analysing the data

[6]:
```
df['Class'].value_counts()
```

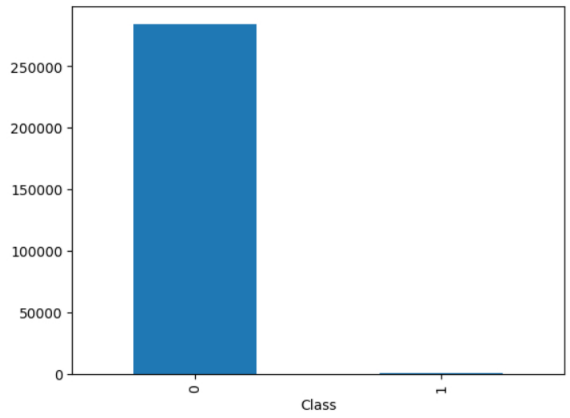
[6]:
```
Class
0    284315
1       492
Name: count, dtype: int64
```

this shows:

1. 492 - Fraud Transactions
2. 284315 - Normal Transactions

[9]:
```
df['Class'].value_counts().plot(kind = 'bar')
```

[9]:
```
<Axes: xlabel='Class'>
```



[10]:
```
fraud = df[df.Class == 1]
normal = df[df.Class ==0]
```

[12]:
```
df.groupby('Class').mean()
```

[12]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	
0	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.000987	0.004467	...	-0.000644	-0.001235	-0.000024	0.000070	0.000182	-0.000072	-0.00
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0.05

2 rows × 30 columns

[13]:
```
fraud.Amount.describe()
```

[13]:
```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

[14]:
```
normal.Amount.describe()
```

[14]:
```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```

Sampling

[15]:
```
normal_sample = normal.sample( n = 492)
```

Type Markdown and LaTeX: α^2

[16]:
```
new = pd.concat([normal_sample,fraud],axis = 0)
```

[17]:
```
new.head()
```

[17]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V
81548	58983.0	1.102351	-0.026205	-0.318137	1.367244	1.818041	4.094025	-0.903076	1.048627	-0.480430	...	-0.011720	-0.337637	-0.016204	0.978174	0.356365	-0.065083	0.0188
248954	154170.0	2.215449	-0.837615	-1.649572	-1.110129	-0.175837	-0.396610	-0.507356	-0.024385	-0.449927	...	0.028578	-0.017165	0.201569	0.250389	-0.046450	-0.296942	-0.0569
147894	89142.0	-0.963780	1.908967	-0.745059	-0.444631	0.210726	-0.763743	-0.021048	-2.297960	-0.187320	...	2.715426	0.650512	0.236859	0.009413	-0.654447	-0.201960	0.4973
103998	68877.0	1.049781	-1.055082	-0.434712	-0.567684	-0.853987	-1.120887	0.122131	-0.322809	-0.927458	...	-0.054519	-0.831863	-0.144316	-0.019150	0.374362	-0.446718	-0.0722
154450	101588.0	1.682079	-1.032718	0.739374	1.982682	-1.255892	1.228025	-1.570709	0.457974	3.897168	...	-0.010427	0.490437	0.076999	0.508633	-0.186304	-0.541639	0.0764

5 rows × 31 columns

[18]:
```
new['Class'].value_counts()
```

[18]:
```
Class
0    492
1    492
Name: count, dtype: int64
```

[19]:
```
new.groupby('Class').mean()
```

[19]:

Class	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V26
0	97978.713415	0.037683	-0.076045	0.050421	-0.024116	0.216070	-0.012280	0.020753	0.031764	0.007637	...	-0.026139	0.015908	0.037603	-0.032827	-0.002142	0.003429	0.021645
1	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	0.372319	0.713588	0.014049	-0.040308	-0.105130	0.041449	0.051648

2 rows × 30 columns

Model building

[20]:
```
X = new.drop(columns = 'Class',axis = 1)
Y = new['Class']
```

[21]:
```
X
```

[21]:

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V20	V21	V22	V23	V24	V25	V
81548	58983.0	1.102351	-0.026205	-0.318137	1.367244	1.818041	4.094025	-0.903076	1.048627	-0.480430	...	0.098909	-0.011720	-0.337637	-0.016204	0.978174	0.356365	-0.0650
248954	154170.0	2.215449	-0.837615	-1.649572	-1.110129	-0.175837	-0.396610	-0.507356	-0.024385	-0.449927	...	-0.134427	0.028578	-0.017165	0.201569	0.250389	-0.046450	-0.2969
147894	89142.0	-0.963780	1.908967	-0.745059	-0.444631	0.210726	-0.763743	-0.021048	-2.297960	-0.187320	...	-0.530877	2.715426	0.650512	0.236859	0.009413	-0.654447	-0.2019
103998	68877.0	1.049781	-1.055082	-0.434712	-0.567684	-0.853987	-1.120887	0.122131	-0.322809	-0.927458	...	0.420904	-0.054519	-0.831863	-0.144316	-0.019150	0.374362	-0.4467
154450	101588.0	1.682079	-1.032718	0.739374	1.982682	-1.255892	1.228025	-1.570709	0.457974	3.897168	...	-0.179489	-0.010427	0.490437	0.076999	0.508633	-0.186304	-0.5416
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
279863	169142.0	-1.927883	1.125653	-4.518331	1.749293	-1.566487	-2.010494	-0.882850	0.697211	-2.064945	...	1.252967	0.778584	-0.319189	0.639419	-0.294885	0.537503	0.7883
280143	169347.0	1.378559	1.289381	-5.004247	1.411850	0.442581	-1.326536	-1.413170	0.248525	-1.127396	...	0.226138	0.370612	0.028234	-0.145640	-0.081049	0.521875	0.7394
280149	169351.0	-0.676143	1.126366	-2.213700	0.468308	-1.120541	-0.003346	-2.234739	1.210158	-0.652250	...	0.247968	0.751826	0.834108	0.190944	0.032070	-0.739695	0.4711
281144	169966.0	-3.113832	0.585864	-5.399730	1.817092	-0.840618	-2.943548	-2.208002	1.058733	-1.632333	...	0.306271	0.583276	-0.269209	-0.456108	-0.183659	-0.328168	0.6061
281674	170348.0	1.991976	0.158476	-2.583441	0.408670	1.151147	-0.096695	0.223050	-0.068384	0.577829	...	-0.017652	-0.164350	-0.295135	-0.072173	-0.450261	0.313267	-0.2896

984 rows × 30 columns

[22]:
```
Y
```

[22]:
```
81548     0
248954    0
147894    0
103998    0
154450    0
         ..
279863    1
280143    1
280149    1
```

```
281144    1
281674    1
Name: Class, Length: 984, dtype: int64
```

[23]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.5,random_state = 10)
```

[29]:
```
model = LogisticRegression()
```

[30]:
```
model.fit(X_train,Y_train)
```

[30]:
```
LogisticRegression
LogisticRegression()
```

[31]:
```
train_pre = model.predict(X_train)
train_acc = accuracy_score(train_pre,Y_train)
```

[32]:
```
train_acc
```

[32]:
```
0.926829268292683
```

[33]:
```
test_pre = model.predict(X_test)
test_acc = accuracy_score(test_pre,Y_test)
```

[34]:
```
test_acc
```

[34]:
```
0.9329268292682927
```

[35]:
```
rate = (test_acc)*100
print(f'{int(rate)}%')
```

```
93%
```

[ ]:
```

```

+ Code + Markdown

View Active Events