FRAUD DETECTION CREDIT CARD

+ Code + Text

```
import pandas as pd
fraud_data = pd.read_csv("/content/creditcard.csv")
fraud_data.head()
```

	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	0.098
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270

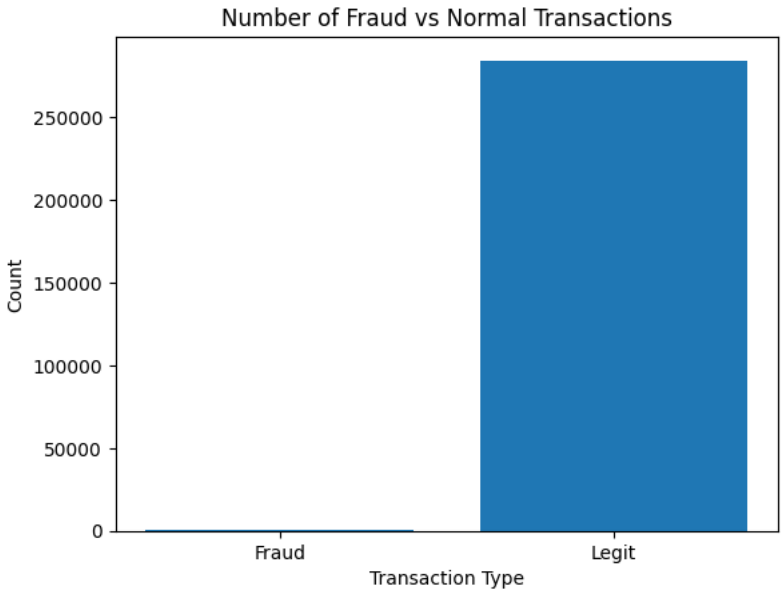
5 rows × 31 columns

```
fraud_data= fraud_data.dropna()
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
# Assuming 'fraud_data' is the DataFrame containing your data
fraud = fraud_data[fraud_data['Class'] == 1]
legit = fraud_data[fraud_data['Class'] == 0]
```

```
import matplotlib.pyplot as plt
fraud_count = len(fraud)
normal_count = len(legit)

# Create a bar plot
plt.bar(['Fraud', 'Legit'], [fraud_count, normal_count])

# Set the plot title and labels
plt.title('Number of Fraud vs Normal Transactions')
plt.xlabel('Transaction Type')
plt.ylabel('Count')

# Display the plot
plt.show()
```




```
fraud.shape
```

```
(492, 31)
```

```
normal.shape
```

```
(284315, 31)
```

```
fraud.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	
count	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	492.000000	...	49
mean	80746.806911	-4.771948	3.623778	-7.033281	4.542029	-3.151225	-1.397737	-5.568731	0.570636	-2.581123	...	
std	47835.365138	6.783687	4.291216	7.110937	2.873318	5.372468	1.858124	7.206773	6.797831	2.500896	...	
min	406.000000	-30.552380	-8.402154	-31.103685	-1.313275	-22.105532	-6.406267	-43.557242	-41.044261	-13.434066	...	-2
25%	41241.500000	-6.036063	1.188226	-8.643489	2.373050	-4.792835	-2.501511	-7.965295	-0.195336	-3.872383	...	
50%	75568.500000	-2.342497	2.717869	-5.075257	4.177147	-1.522962	-1.424616	-3.034402	0.621508	-2.208768	...	
75%	128483.000000	-0.419200	4.971257	-2.276185	6.348729	0.214562	-0.413216	-0.945954	1.764879	-0.787850	...	
max	170348.000000	2.132386	22.057729	2.250210	12.114672	11.095089	6.474115	5.802537	20.007208	3.353525	...	2

8 rows × 31 columns

```
legit.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	
count	284315.000000	284315.000000	284315.000000	284315.000000	284315.000000	284315.000000	284315.000000	284315.000000	284315.00
mean	94838.202258	0.008258	-0.006271	0.012171	-0.007860	0.005453	0.002419	0.009637	-0.00
std	47484.015786	1.929814	1.636146	1.459429	1.399333	1.356952	1.329913	1.178812	1.16
min	0.000000	-56.407510	-72.715728	-48.325589	-5.683171	-113.743307	-26.160506	-31.764946	-73.21
25%	54230.000000	-0.917544	-0.599473	-0.884541	-0.850077	-0.689398	-0.766847	-0.551442	-0.20
50%	84711.000000	0.020023	0.064070	0.182158	-0.022405	-0.053457	-0.273123	0.041138	0.02
75%	139333.000000	1.316218	0.800446	1.028372	0.737624	0.612181	0.399619	0.571019	0.32
max	172792.000000	2.454930	18.902453	9.382558	16.875344	34.801666	73.301626	120.589494	18.70

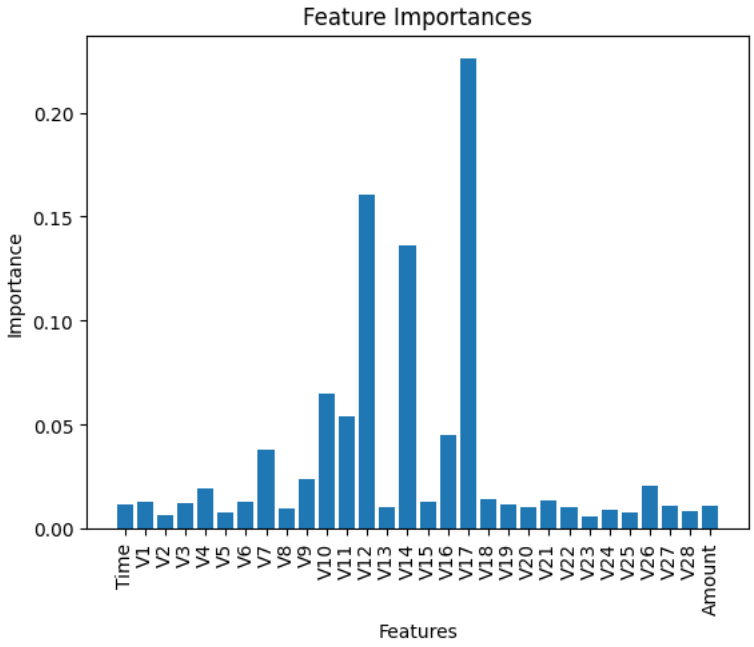
8 rows × 31 columns

```
x=fraud_data.drop(['Class'],axis=1)
y=fraud_data['Class']


from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

# Split the features and target variables

x=fraud_data.drop(['Class'],axis=1)
y=fraud_data['Class']


# Split the data into training and test sets
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

# Create a Random Forest Classifier
rf = RandomForestClassifier(n_estimators=150, max_depth=None, max_features=10, random_state=42)

# Train the classifier on the training data
rf.fit(x_train, y_train)

# Evaluate the accuracy on the entire dataset
accuracy = rf.score(x, y)
print("Accuracy on the entire dataset:", accuracy)

# Evaluate the accuracy on the test set
test_accuracy = rf.score(x_test, y_test)
```

```
print("Accuracy on the test set:", test_accuracy)

# Evaluate the accuracy on the training set
train_accuracy = rf.score(x_train, y_train)
print("Accuracy on the training set:", train_accuracy)
```

```
Accuracy on the entire dataset: 0.9999157324082625
Accuracy on the test set: 0.9995962220427653
Accuracy on the training set: 0.9999956110513727
```

```
# Access feature importances
importances = rf.feature_importances_

# Get column names from the fraud_data DataFrame
feature_names = fraud_data.columns[:-1]  # Exclude the last column 'Class'

# Create a bar plot
plt.bar(feature_names, importances)

# Set the plot title and labels
plt.title('Feature Importances')
plt.xlabel('Features')
plt.ylabel('Importance')

# Rotate x-axis labels for better readability (optional)
plt.xticks(rotation=90)

# Display the plot
plt.show()
```



✓ 0s completed at 01:31