2. Describe your fraud detection model in elaboration.:

```
import pandas as pd
df=pd.read_csv("/content/Fraud.csv")
```

```
x =df[["amount"]].values
x
```

```
array([[9.83964000e+03],
       [1.86428000e+03],
       [1.81000000e+02],
       ...,
       [6.31140928e+06],
       [8.50002520e+05],
       [8.50002520e+05]])
```

```
y =df[["isFraud"]].values   #dependent
y
```

```
array([[0],
       [0],
       [1],
       ...,
       [1],
       [1],
       [1]])
```

```
import numpy as np
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(x)
x=imputer.transform(x)
x
```

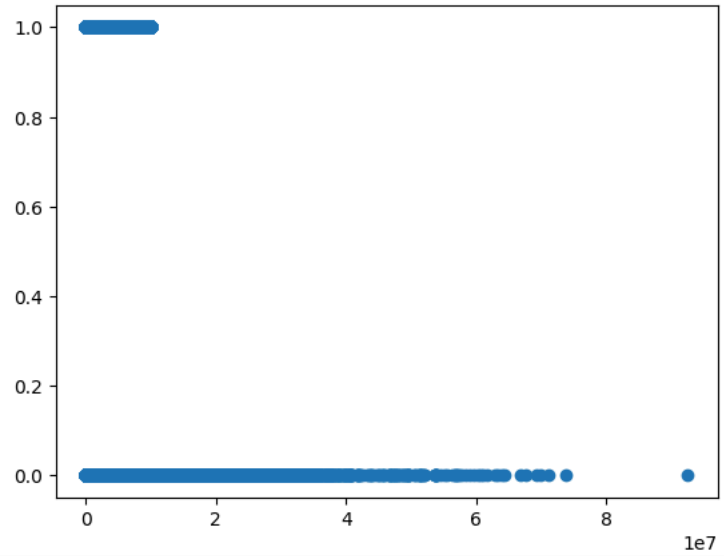
```
array([[9.83964000e+03],
       [1.86428000e+03],
       [1.81000000e+02],
       ...,
       [6.31140928e+06],
       [8.50002520e+05],
       [8.50002520e+05]])
```

```
import numpy as np
from sklearn.impute import SimpleImputer
imputer=SimpleImputer(missing_values=np.nan,strategy='mean')
imputer.fit(y)
y=imputer.transform(y)
y
```

```
array([[0.],
       [0.],
       [1.],
       ...,
       [1.],
       [1.],
       [1.]])
```

```
import matplotlib.pyplot as plt
plt.scatter(x="amount",y="isFraud",data=df)
```

```
<matplotlib.collections.PathCollection at 0x7d98127eba00>
```

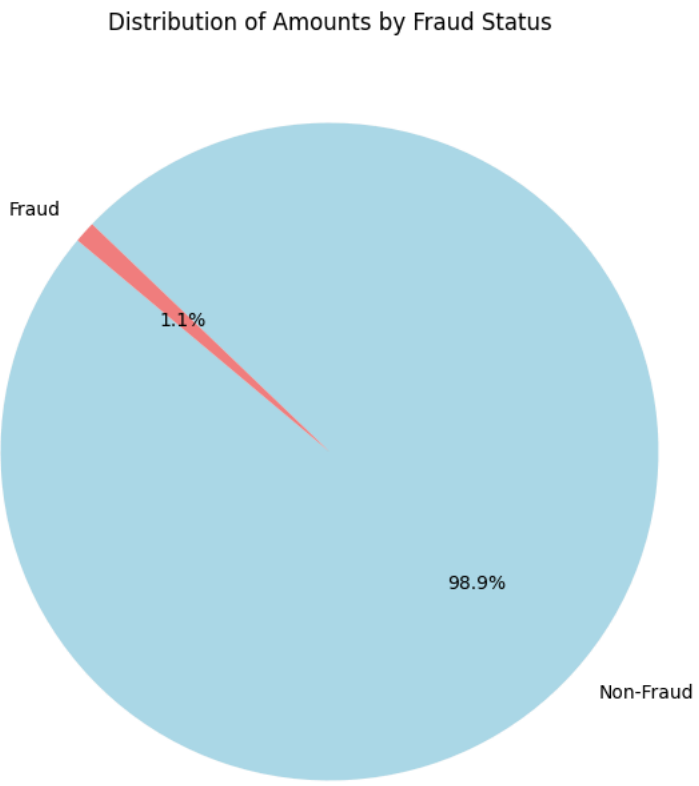



```
import matplotlib.pyplot as plt
import pandas as pd

# Sample data
#
df = pd.DataFrame(df)

# Aggregate the data
fraud_sum = df[df["isFraud"] == 1]["amount"].sum()
non_fraud_sum = df[df['isFraud'] == 0]['amount'].sum()
sizes = [non_fraud_sum, fraud_sum]
labels = ['Non-Fraud', 'Fraud']
colors = ['lightblue', 'lightcoral']

# Create a pie chart
plt.figure(figsize=(8, 8))
plt.pie(sizes, labels=labels, colors=colors, autopct='%1.1f%%', startangle=140)
plt.title('Distribution of Amounts by Fraud Status')
plt.show()
```



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df[["amount"]], df[["isFraud"]], test_size=0.2,)
```

```
x_train.values
```

```
array([[ 71938.28],
       [  3730.85],
       [186306.9 ],
       ...,
       [ 87039.05],
       [306032.01],
       [124804.72]])
```

```
x_test.values
```

```
array([[7.0750850e+04],
       [4.6861704e+05],
       [8.1910000e+01],
       ...,
       [3.0134300e+03],
       [6.7375190e+04],
       [9.5611200e+03]])
```

```
from sklearn.linear_model import LogisticRegression
model=LogisticRegression()
model.fit(x_train,y_train)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:1408: DataConversionWarning: A column-vector y was passed when a
  y = column_or_1d(y, warn=True)
```

▾ LogisticRegression

LogisticRegression()

```
model.predict(x_test)
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
model.predict(x_train)
```

```
array([0, 0, 0, ..., 0, 0, 0])
```

```
model.predict([[2]])
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Logi
  warnings.warn(
array([0])
```

```
# from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.ensemble import RandomForestClassifier
import pandas as pd

# # Example DataFrame (replace with your actual data)
# data = {
#     'amount': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100],
#     'isFraud': [0, 0, 1, 0, 1, 1, 0, 1, 0, 1]
# }
# df = pd.DataFrame(data)

# Splitting the data into training and testing sets
X = df[['amount']]
y = df['isFraud']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Training the model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Making predictions
y_pred = model.predict(X_test)

# Calculating the accuracy
accuracy = accuracy_score(y_test, y_pred)

# Printing the accuracy
print(f"Model Accuracy: {accuracy * 100:.2f}%")
```

```
Model Accuracy: 99.87%
```