Upload the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving manoj.csv to manoj.csv
```

Load the Dataset

```
import pandas as pd

# Load the dataset
df = pd.read_csv("manoj.csv")
df.head()
```

	CustomerID	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	C
0	15776156	1	22.08	11.46	2	4	4	1.585	0	0	0	1	2	100	1213	
1	15739548	0	22.67	7.00	2	8	4	0.165	0	0	0	0	2	160	1	
2	15662854	0	29.58	1.75	1	4	4	1.250	0	0	0	1	2	280	1	
3	15687688	0	21.67	11.50	1	5	3	0.000	1	1	11	1	2	0	1	
4	15715750	1	20.17	8.17	2	6	4	1.960	1	1	14	0	2	60	159	

Data Exploration

```
print("Shape:", df.shape)
print("Columns:", df.columns.tolist())
df.info()
df.describe()
```

```
Shape: (690, 16)
Columns: ['CustomerID', 'A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 690 entries, 0 to 689
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   CustomerID  690 non-null    int64
 1   A1          690 non-null    int64
 2   A2          690 non-null    float64
 3   A3          690 non-null    float64
 4   A4          690 non-null    int64
 5   A5          690 non-null    int64
 6   A6          690 non-null    int64
 7   A7          690 non-null    float64
 8   A8          690 non-null    int64
 9   A9          690 non-null    int64
 10  A10         690 non-null    int64
 11  A11         690 non-null    int64
 12  A12         690 non-null    int64
 13  A13         690 non-null    int64
 14  A14         690 non-null    int64
 15  Class       690 non-null    int64
dtypes: float64(3), int64(13)
memory usage: 86.4 KB
```

	CustomerID	A1	A2	A3	A4	A5	
count	6.900000e+02	690.000000	690.000000	690.000000	690.000000	690.000000	690.(
mean	1.569047e+07	0.678261	31.568203	4.758725	1.766667	7.372464	4.(
std	7.150647e+04	0.467482	11.853273	4.978163	0.430063	3.683265	1.
min	1.556571e+07	0.000000	13.750000	0.000000	1.000000	1.000000	1.(
25%	1.563169e+07	0.000000	22.670000	1.000000	2.000000	4.000000	4.(
50%	1.569016e+07	1.000000	28.625000	2.750000	2.000000	8.000000	4.(
75%	1.575190e+07	1.000000	37.707500	7.207500	2.000000	10.000000	5.(
max	1.581544e+07	1.000000	80.250000	28.000000	3.000000	14.000000	9.(

Check for Missing Values and Duplicates

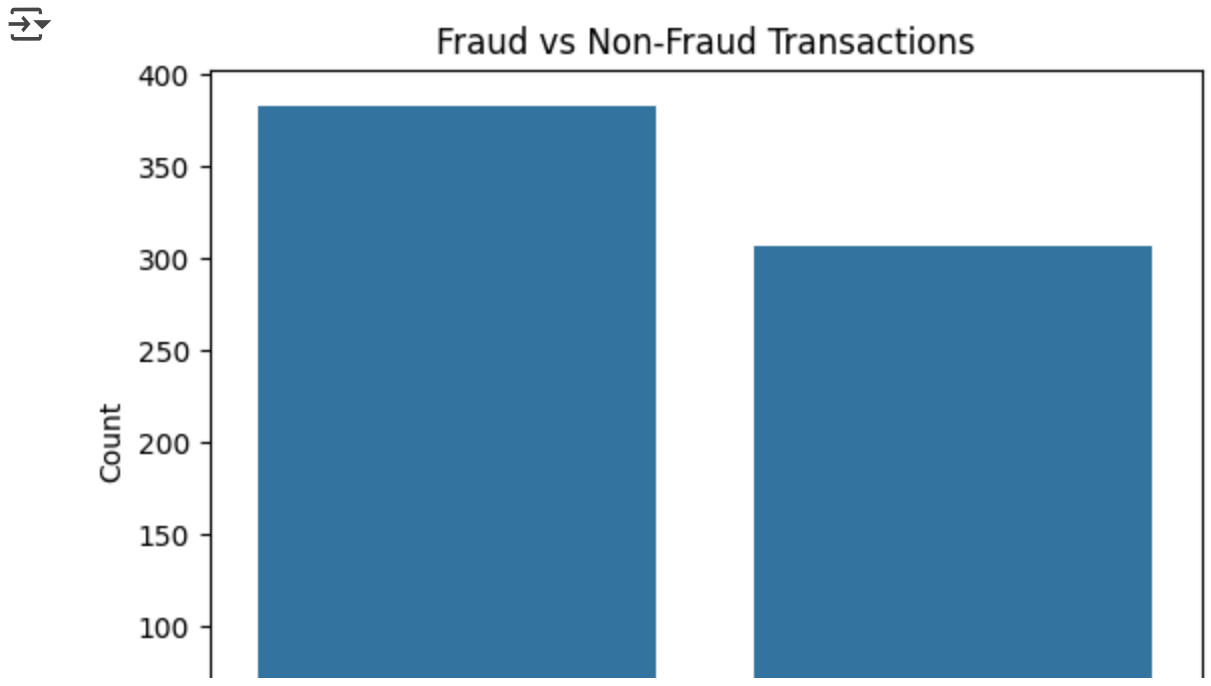
```
# Check missing values
print("Missing values:\n", df.isnull().sum())

# Check for duplicates
print("Duplicate rows:", df.duplicated().sum())
```

```
Missing values:
 CustomerID    0
A1             0
```

```
A2              0
A3              0
A4              0
A5              0
A6              0
A7              0
A8              0
A9              0
A10             0
A11             0
A12             0
A13             0
A14             0
Class           0
dtype: int64
Duplicate rows: 0
```

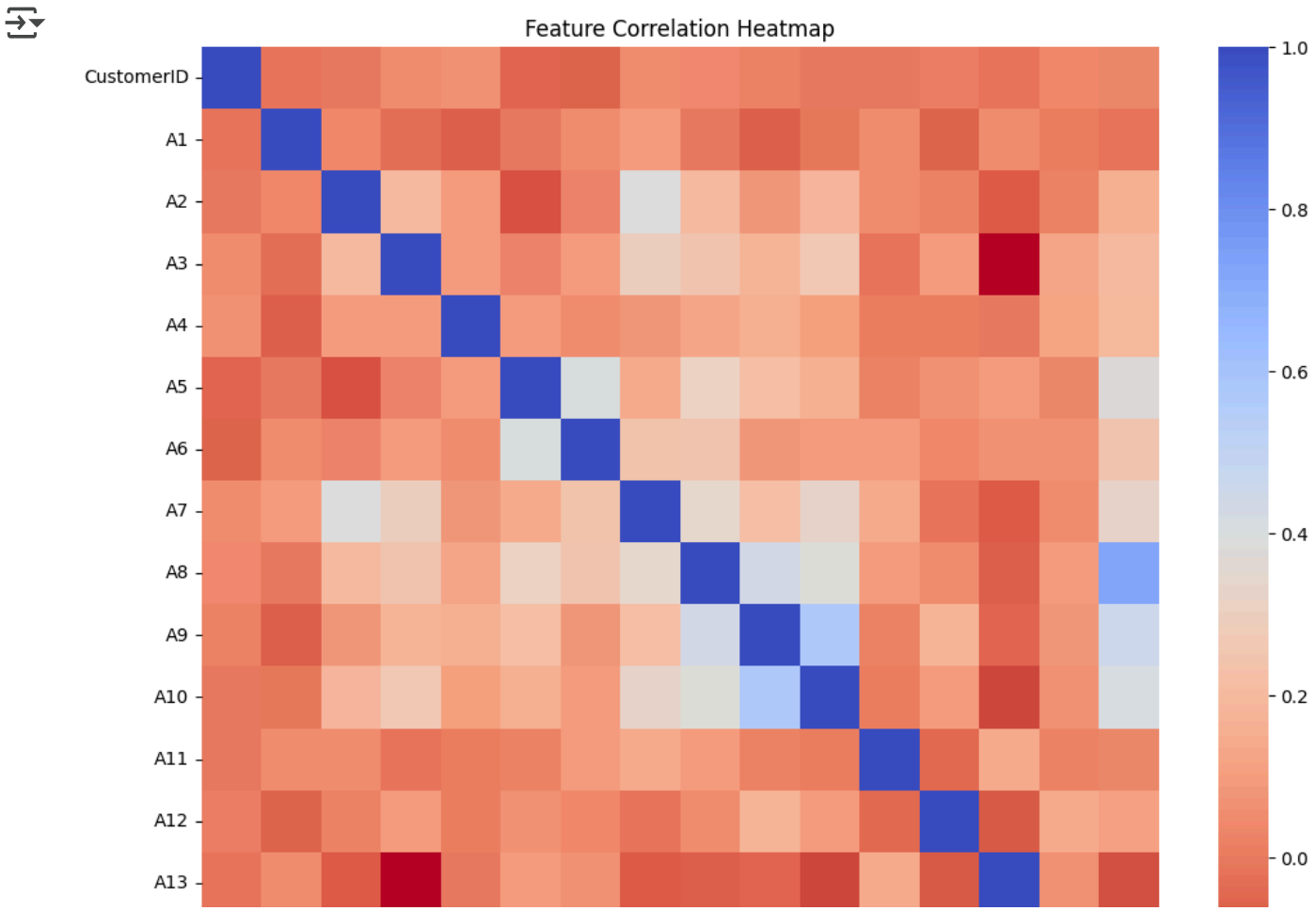
Visualize a Few Features

```
import seaborn as sns
import matplotlib.pyplot as plt

# Countplot for fraud vs non-fraud
sns.countplot(x='Class', data=df)
plt.title("Fraud vs Non-Fraud Transactions")
plt.xlabel("Class (0 = Non-Fraud, 1 = Fraud)")
plt.ylabel("Count")
plt.show()

# Percentage
fraud_percentage = df['Class'].value_counts(normalize=True) * 100
print("Fraud Class Percentage:\n", fraud_percentage)
```



Correlation Matrix

```
plt.figure(figsize=(12, 10))
sns.heatmap(df.corr(), cmap='coolwarm_r', annot=False)
plt.title("Feature Correlation Heatmap")
plt.show()

# Features most correlated with fraud
print(df.corr()['Class'].sort_values(ascending=False).head(10))
```



Feature Scaling & Train-Test Split

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

X = df.drop('Class', axis=1)
y = df['Class']

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_stat
```

Train a Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

model = RandomForestClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Model Evaluation

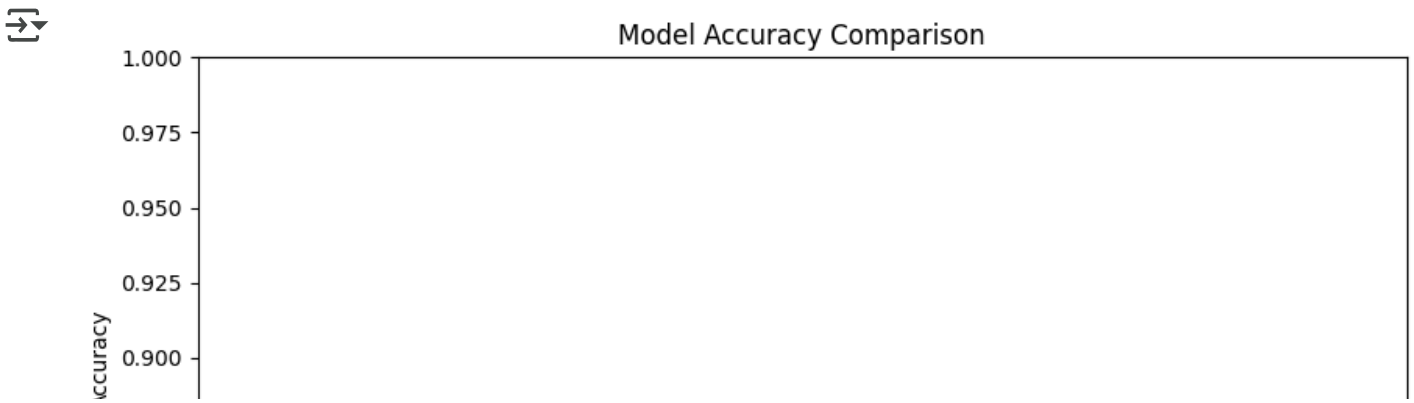
```
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8913043478260869

Confusion Matrix:
 [[81  6]
 [ 9 42]]

Classification Report:
              precision    recall  f1-score   support
```

```
              0       0.90      0.93      0.92        87
              1       0.88      0.82      0.85        51

       accuracy                           0.89       138
      macro avg       0.89      0.88      0.88       138
   weighted avg       0.89      0.89      0.89       138
```

Compare Multiple Models

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier

models = {
    "Logistic Regression": LogisticRegression(),
    "Random Forest": RandomForestClassifier(),
    "SVM": SVC(),
    "Naive Bayes": GaussianNB(),
    "KNN": KNeighborsClassifier()
}

results = {}

for name, clf in models.items():
    clf.fit(X_train, y_train)
    pred = clf.predict(X_test)
    results[name] = accuracy_score(y_test, pred)

# Visualize model comparison
plt.figure(figsize=(10,5))
sns.barplot(x=list(results.keys()), y=list(results.values()))
plt.title("Model Accuracy Comparison")
plt.ylabel("Accuracy")
plt.ylim(0.8, 1.0)
plt.xticks(rotation=45)
plt.show()
```



```
! pip install gradio
```

```
Collecting gradio
  Downloading gradio-5.29.1-py3-none-any.whl.metadata (16 kB)
Collecting aiofiles<25.0,>=22.0 (from gradio)
  Downloading aiofiles-24.1.0-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: anyio<5.0,>=3.0 in /usr/local/lib/python3.11/dist
Collecting fastapi<1.0,>=0.115.2 (from gradio)
  Downloading fastapi-0.115.12-py3-none-any.whl.metadata (27 kB)
Collecting ffmpy (from gradio)
  Downloading ffmpy-0.5.0-py3-none-any.whl.metadata (3.0 kB)
Collecting gradio-client==1.10.1 (from gradio)
  Downloading gradio_client-1.10.1-py3-none-any.whl.metadata (7.1 kB)
Collecting groovy~=0.1 (from gradio)
  Downloading groovy-0.1.2-py3-none-any.whl.metadata (6.1 kB)
Requirement already satisfied: httpx>=0.24.1 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: huggingface-hub>=0.28.1 in /usr/local/lib/python3
Requirement already satisfied: jinja2<4.0 in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: markupsafe<4.0,>=2.0 in /usr/local/lib/python3.11
Requirement already satisfied: numpy<3.0,>=1.0 in /usr/local/lib/python3.11/dist
Requirement already satisfied: orjson~=3.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packa
Requirement already satisfied: pandas<3.0,>=1.0 in /usr/local/lib/python3.11/dis
Requirement already satisfied: pillow<12.0,>=8.0 in /usr/local/lib/python3.11/di
Requirement already satisfied: pydantic<2.12,>=2.0 in /usr/local/lib/python3.11/
Collecting pydub (from gradio)
  Downloading pydub-0.25.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting python-multipart>=0.0.18 (from gradio)
  Downloading python_multipart-0.0.20-py3-none-any.whl.metadata (1.8 kB)
Requirement already satisfied: pyyaml<7.0,>=5.0 in /usr/local/lib/python3.11/dis
Collecting ruff>=0.9.3 (from gradio)
  Downloading ruff-0.11.10-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.w
Collecting safehttpx<0.2.0,>=0.1.6 (from gradio)
  Downloading safehttpx-0.1.6-py3-none-any.whl.metadata (4.2 kB)
Collecting semantic-version~=2.0 (from gradio)
  Downloading semantic_version-2.10.0-py2.py3-none-any.whl.metadata (9.7 kB)
Collecting starlette<1.0,>=0.40.0 (from gradio)
  Downloading starlette-0.46.2-py3-none-any.whl.metadata (6.2 kB)
Collecting tomlkit<0.14.0,>=0.12.0 (from gradio)
  Downloading tomlkit-0.13.2-py3-none-any.whl.metadata (2.7 kB)
```

```
     Requirement already satisfied: typer<1.0,>=0.12 in /usr/local/lib/python3.11/dis
     Requirement already satisfied: typing-extensions~=4.0 in /usr/local/lib/python3.
     Collecting uvicorn>=0.14.0 (from gradio)
       Downloading uvicorn-0.34.2-py3-none-any.whl.metadata (6.5 kB)
     Requirement already satisfied: fsspec in /usr/local/lib/python3.11/dist-packages
     Requirement already satisfied: websockets<16.0,>=10.0 in /usr/local/lib/python3.
     Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.11/dist-packa
     Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-pa
     Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-package
     Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-p
     Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packa
     Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packag
     Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packag
     Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-pa
     Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
     Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pa
     Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
     Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
     Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.1
     Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python
```

```
import gradio as gr
import numpy as np

# Get feature names
feature_names = df.drop('Class', axis=1).columns.tolist()

def predict_fraud(*inputs):
    # Convert to numpy array
    input_array = np.array(inputs).reshape(1, -1)

    # Scale the input using the same scaler
    input_scaled = scaler.transform(input_array)

    # Predict
    prediction = model.predict(input_scaled)[0]
    probability = model.predict_proba(input_scaled)[0][1]

    result = "🚨 Fraudulent Transaction" if prediction == 1 else "✅ Legitimate Transaction
    confidence = f"Confidence: {round(probability * 100, 2)}%"

    return f"{result}\n{confidence}"
# Create input components dynamically based on feature columns
inputs = [gr.Number(label=col) for col in feature_names]
output = gr.Textbox(label="Prediction Result")

# Launch the app
gr.Interface(
    fn=predict_fraud,
    inputs=inputs,
    outputs=output,
    title="💳 AI Credit Card Fraud Detector",
    description="Enter transaction details to check if it's fraud or legitimate.",
    theme="default"
).launch()
```

```
It looks like you are running Gradio on a hosted a Jupyter notebook. For the Gra

Colab notebook detected. To show errors in colab notebook, set debug=True in lau
* Running on public URL: https://f4c330a4e19fba9cd5.gradio.live

This share link expires in 1 week. For free permanent hosting and GPU upgrades,
```

No interface is running right now