In [19]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import AdaBoostClassifier,GradientBoostingClassifier,RandomFores
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score,confusion_matrix,f1_score,recall_score,prec
from sklearn.preprocessing import LabelEncoder,StandardScaler
```

In [17]:
```
data=pd.read_csv("Bank_Transaction_Fraud_Detection.csv")
data.head(6)
```

Out[17]:

	Customer_ID	Customer_Name	Gender	Age	State	City	Bank_Branch
0	d5f6ec07-d69e-4f47-b9b4-7c58ff17c19e	Osha Tella	Male	60	Kerala	Thiruvananthapuram	Thiruvananthapuram Branch
1	7c14ad51-781a-4db9-b7bd-67439c175262	Hredhaan Khosla	Female	51	Maharashtra	Nashik	Nashik Branch
2	3a73a0e5-d4da-45aa-85f3-528413900a35	Ekani Nazareth	Male	20	Bihar	Bhagalpur	Bhagalpur Branch
3	7902f4ef-9050-4a79-857d-9c2ea3181940	Yamini Ramachandran	Female	57	Tamil Nadu	Chennai	Chennai Branch
4	3a4bba70-d9a9-4c5f-8b92-1735fd8c19e9	Kritika Rege	Female	43	Punjab	Amritsar	Amritsar Branch
5	6c870d65-76b0-431d-bdf3-9292998e8211	Ishanvi Dar	Male	54	Gujarat	Ahmedabad	Ahmedabad Branch

6 rows × 24 columns

In [18]:
```
data=data.drop(["Customer_ID","Customer_Name","Transaction_ID","Customer_Contact","Cus
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 17 columns):
 #   Column                  Non-Null Count   Dtype  
---  ------                  --------------   -----  
 0   Gender                  200000 non-null  object 
 1   Age                     200000 non-null  int64  
 2   State                   200000 non-null  object 
 3   City                    200000 non-null  object 
 4   Bank_Branch             200000 non-null  object 
 5   Account_Type            200000 non-null  object 
 6   Transaction_Date        200000 non-null  object 
 7   Transaction_Time        200000 non-null  object 
 8   Transaction_Amount      200000 non-null  float64
 9   Transaction_Type        200000 non-null  object 
 10  Merchant_Category       200000 non-null  object 
 11  Account_Balance         200000 non-null  float64
 12  Transaction_Device      200000 non-null  object 
 13  Transaction_Location    200000 non-null  object 
 14  Device_Type             200000 non-null  object 
 15  Is_Fraud                200000 non-null  int64  
 16  Transaction_Description  200000 non-null  object 
dtypes: float64(2), int64(2), object(13)
memory usage: 25.9+ MB
```

In [4]:
```
data.describe()
```

Out[4]:

	Age	Transaction_Amount	Account_Balance	Is_Fraud
count	200000.000000	200000.000000	200000.000000	200000.000000
mean	44.015110	49538.015554	52437.988784	0.050440
std	15.288774	28551.874004	27399.507128	0.218852
min	18.000000	10.290000	5000.820000	0.000000
25%	31.000000	24851.345000	28742.395000	0.000000
50%	44.000000	49502.440000	52372.555000	0.000000
75%	57.000000	74314.625000	76147.670000	0.000000
max	70.000000	98999.980000	99999.950000	1.000000

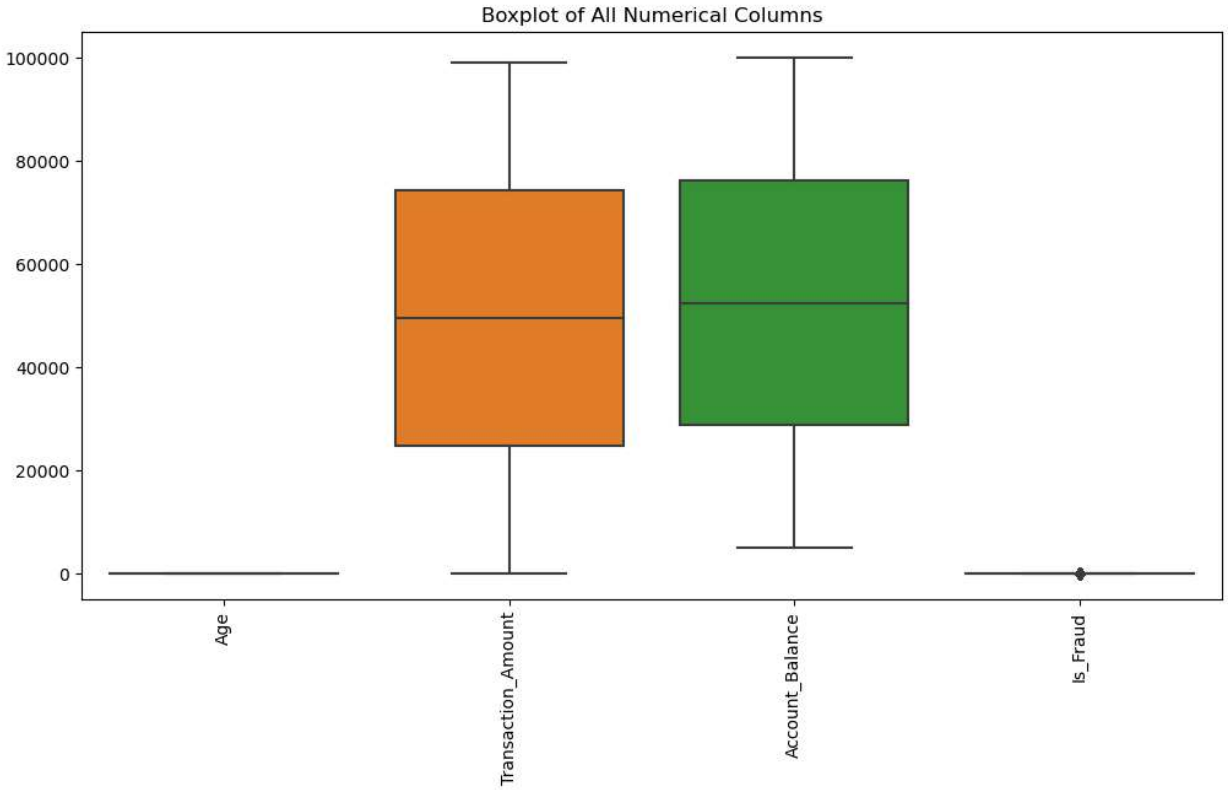
In [5]:
```
data.isnull().sum()
```

Out[5]:
```
Gender                      0
Age                         0
State                       0
City                        0
Bank_Branch                 0
Account_Type                0
Transaction_Date            0
Transaction_Time            0
Transaction_Amount          0
Transaction_Type            0
Merchant_Category           0
Account_Balance             0
Transaction_Device          0
Transaction_Location        0
Device_Type                 0
Is_Fraud                    0
Transaction_Description     0
dtype: int64
```

In [6]:
```
data.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
numerical_cols = data.select_dtypes(include=['number'])
plt.figure(figsize=(12, 6))
sns.boxplot(data=numerical_cols)
plt.xticks(rotation=90)
plt.title("Boxplot of All Numerical Columns")
plt.show()
```



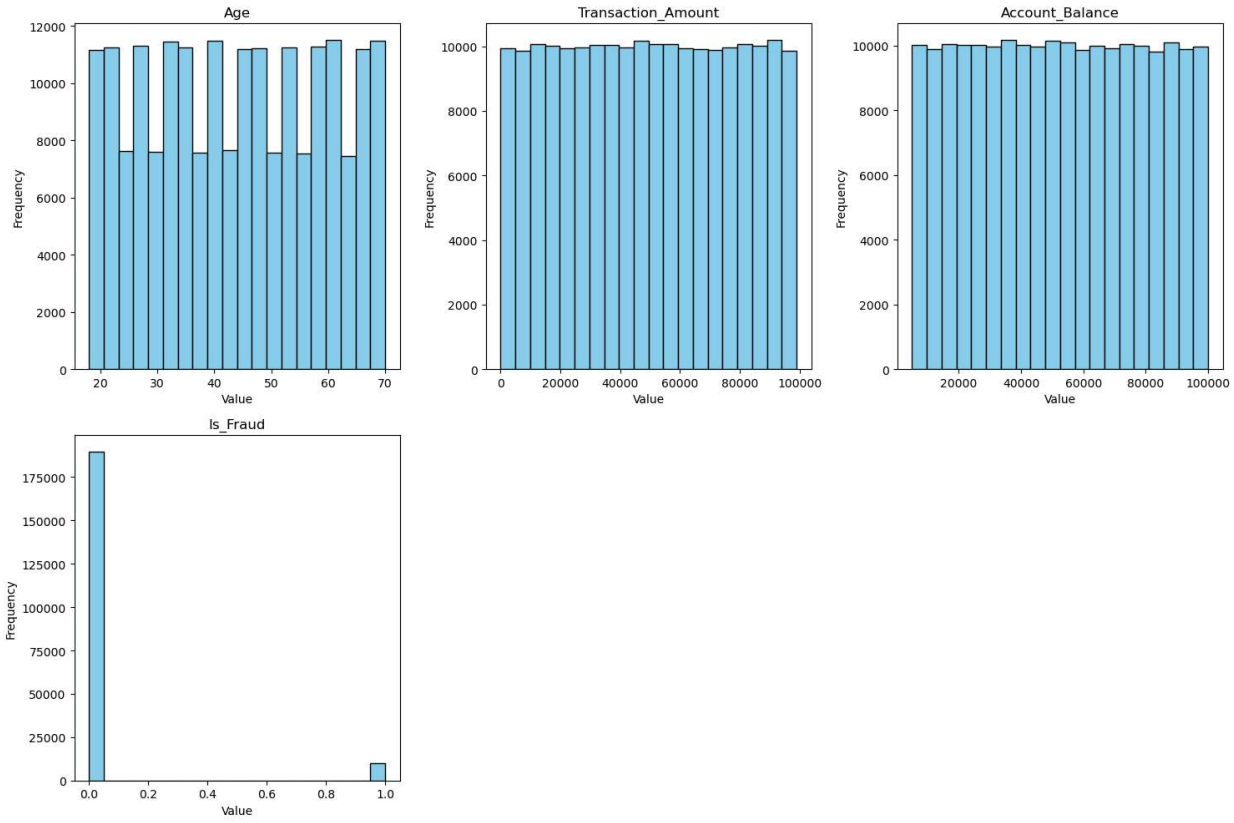
In [8]:
```
numerical_cols = data.select_dtypes(include=['number'])

num_cols = len(numerical_cols.columns)
fig, axes = plt.subplots(nrows=num_cols // 3 + 1, ncols=3, figsize=(15, num_cols * 2.5
```

```
axes = axes.flatten()
for i, col in enumerate(numerical_cols.columns):
    axes[i].hist(data[col], bins=20, color='skyblue', edgecolor='black')
    axes[i].set_title(col)
    axes[i].set_xlabel("Value")
    axes[i].set_ylabel("Frequency")

for i in range(num_cols, len(axes)):
    fig.delaxes(axes[i])

plt.tight_layout()
plt.show()
```



In [9]:
```
categorical_cols = data.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le
```

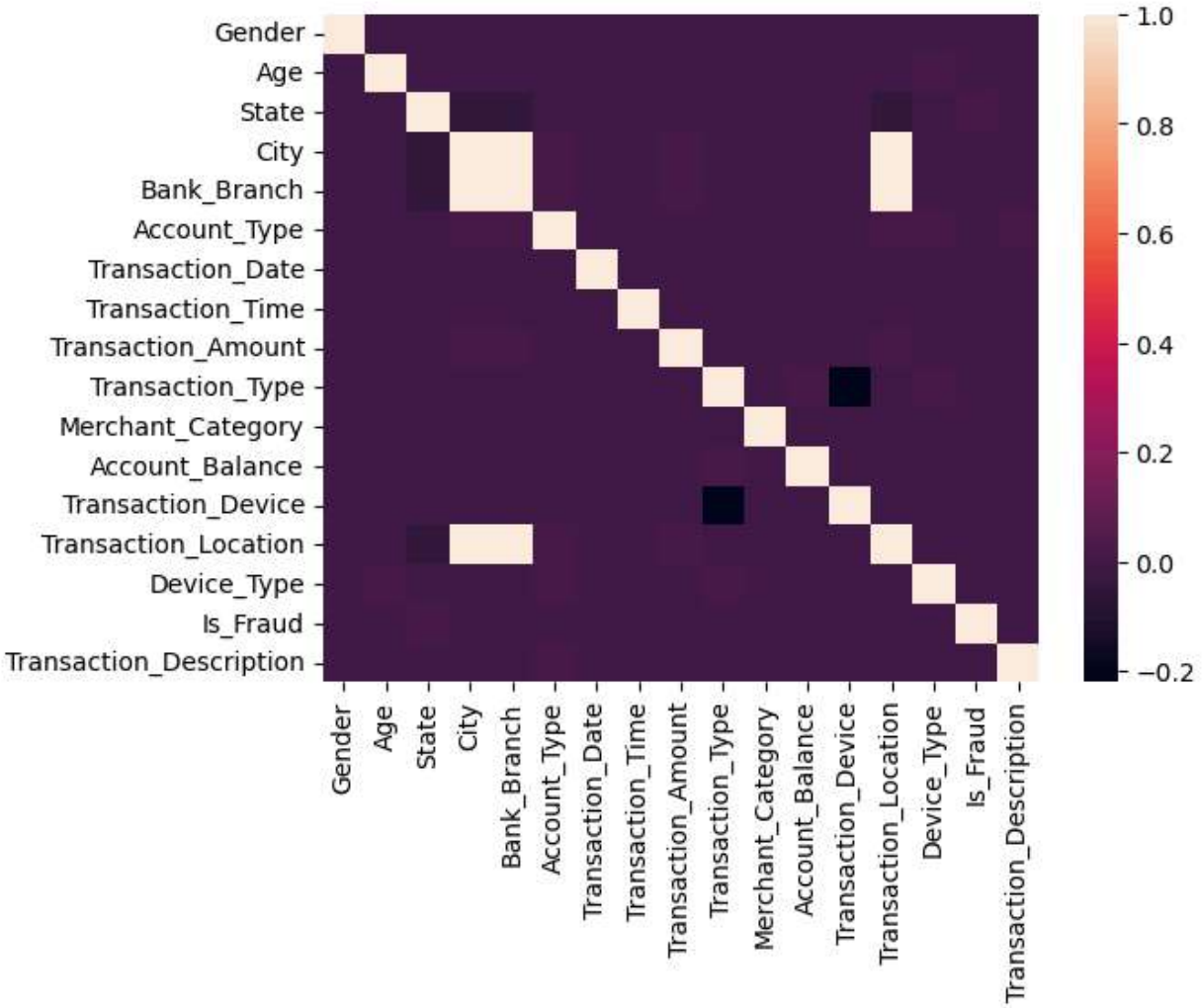
In [10]:
```
data.head(5)
```

Out[10]:

	Gender	Age	State	City	Bank_Branch	Account_Type	Transaction_Date	Transaction_Time	Transac
0	1	60	15	127	127	2	22	52151	
1	0	51	18	100	100	0	10	55958	
2	1	20	4	13	13	2	24	10257	
3	0	57	28	22	22	0	18	40370	
4	0	43	25	7	7	2	29	60052	

In [11]:
```
sns.heatmap(data.corr())
```

Out[11]:
```
<Axes: >
```



In [12]:
```
data.corr()
```

Out[12]:

	Gender	Age	State	City	Bank_Branch	Account_Type	Transa
Gender	1.000000	0.001692	0.002899	-0.000299	-0.000299	0.001260	
Age	0.001692	1.000000	-0.004638	-0.002284	-0.002284	-0.000281	
State	0.002899	-0.004638	1.000000	-0.046278	-0.046278	-0.002314	
City	-0.000299	-0.002284	-0.046278	1.000000	1.000000	0.007245	
Bank_Branch	-0.000299	-0.002284	-0.046278	1.000000	1.000000	0.007245	
Account_Type	0.001260	-0.000281	-0.002314	0.007245	0.007245	1.000000	
Transaction_Date	-0.003731	-0.001401	0.002415	0.002291	0.002291	0.002025	
Transaction_Time	-0.001941	-0.001294	0.000640	0.000812	0.000812	-0.001731	
Transaction_Amount	0.001468	-0.003087	0.002480	0.005674	0.005674	-0.004737	
Transaction_Type	0.001339	-0.001291	-0.000130	0.000101	0.000101	0.001876	
Merchant_Category	0.002233	-0.000381	0.001078	-0.001047	-0.001047	0.000169	
Account_Balance	-0.000392	0.000269	0.000136	-0.002628	-0.002628	-0.001506	
Transaction_Device	0.002109	0.002429	0.001103	-0.001441	-0.001441	-0.003000	
Transaction_Location	-0.000261	-0.002291	-0.044936	0.999910	0.999910	0.007216	
Device_Type	-0.000534	0.004997	-0.001835	-0.000052	-0.000052	0.004613	
Is_Fraud	0.000649	-0.001517	0.005716	0.002800	0.002800	-0.002592	
Transaction_Description	-0.000869	-0.000346	0.001027	0.001547	0.001547	0.005007	

In [13]:
```
models={
    "Random Forest Classifier":RandomForestClassifier(),
    "Decision Tree Classifier":DecisionTreeClassifier(),
    "Ada Boost Classifier":AdaBoostClassifier(),
    "Logistic Regression":LogisticRegression(),
    "Gradiant Boosting Classifier":GradientBoostingClassifier(),
    "Naive Bayes Classifier":GaussianNB()
}
```

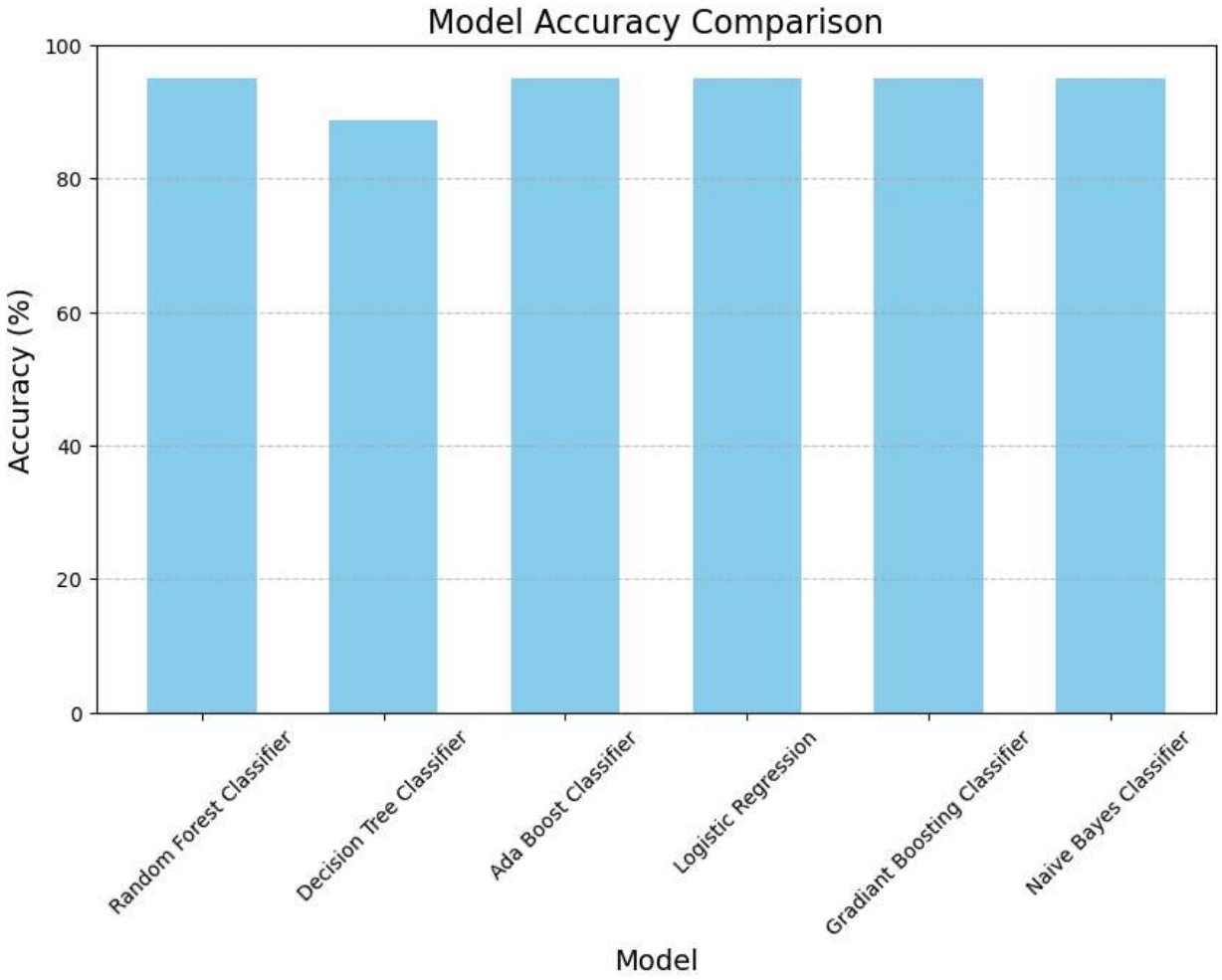
In [14]:
```
x=data.drop(["Is_Fraud"],axis=1)
y=data["Is_Fraud"]
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=42)
ss=StandardScaler()
x_train=ss.fit_transform(x_train)
x_test=ss.transform(x_test)
```

In [15]:
```
metrics = {
    'Model': [],
    'Accuracy': [],
    'Precision': [],
    'Recall':[]
}
for name,model in models.items():
    model.fit(x_train,y_train)
    y_pred=model.predict(x_test)
```

```
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    recall=recall_score(y_test,y_pred)
    precision=precision_score(y_test,y_pred)
    metrics['Model'].append(name)
    metrics['Accuracy'].append(acc * 100)
    metrics['Precision'].append(precision * 100)
    metrics['Recall'].append(recall * 100)
    print(f"Model: {name}")
    print(f"Accuracy: {acc*100}")
    print("-" * 30)
```

```
C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Model: Random Forest Classifier
Accuracy: 94.88600000000001
------------------------------
Model: Decision Tree Classifier
Accuracy: 88.69200000000001
------------------------------
Model: Ada Boost Classifier
Accuracy: 94.882
------------------------------
Model: Logistic Regression
Accuracy: 94.88600000000001
------------------------------
C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
Model: Gradiant Boosting Classifier
Accuracy: 94.884
------------------------------
Model: Naive Bayes Classifier
Accuracy: 94.88600000000001
------------------------------
C:\Users\lenovo\anaconda3\Lib\site-packages\sklearn\metrics\_classification.py:1469:
UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 due to no predi
cted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [16]:
```
metrics_df = pd.DataFrame(metrics)
plt.figure(figsize=(10, 6))
plt.bar(metrics_df['Model'], metrics_df['Accuracy'], color='skyblue', width=0.6)
plt.title('Model Accuracy Comparison', fontsize=16)
plt.ylabel('Accuracy (%)', fontsize=14)
plt.xlabel('Model', fontsize=14)
plt.xticks(rotation=45)
plt.ylim(0, 100)
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```


Model Accuracy Comparison
Accuracy (%)
100
80
60
40
20
0
Random Forest Classifier
Decision Tree Classifier
Ada Boost Classifier
Logistic Regression
Gradiant Boosting Classifier
Naive Bayes Classifier
Model