## Dataset Import

```
import pandas as pd
from google.colab import files
uploaded = files.upload()
df = pd.read_csv("sample.csv")
print(df.head())
```

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 sample.csv to sample (4).csv
   tweet_id     author_id  inbound                      created_at  \
0    119237        105834     True  Wed Oct 11 06:55:44 +0000 2017
1    119238  ChaseSupport    False  Wed Oct 11 13:25:49 +0000 2017
2    119239        105835     True  Wed Oct 11 13:00:09 +0000 2017
3    119240  VirginTrains    False  Tue Oct 10 15:16:08 +0000 2017
4    119241        105836     True  Tue Oct 10 15:17:21 +0000 2017

                                                text response_tweet_id  \
0  @AppleSupport causing the reply to be disregar...            119236
1  @105835 Your business means a lot to us. Pleas...               NaN
2  @76328 I really hope you all change but I'm su...            119238
3  @105836 LiveChat is online at the moment - htt...            119241
4  @VirginTrains see attached error message. I've...            119243

   in_response_to_tweet_id
0                      NaN
1                 119239.0
2                      NaN
3                 119242.0
4                 119240.0
```

## Data Preprocessing

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder

# Sample raw dataset
data = {
    'Chat_ID': [1001, 1002, 1003, 1004, 1005, 1002],  # Duplicate
    'User_Region': ['East', 'West', 'South', 'East', 'North', 'West'],
    'Issue_Type': ['Billing', 'Technical', None, 'Billing', 'Technical', 'Technical'],
    'Chat_Duration': [320, np.nan, 2000, 295, 110, np.nan],
    'Response_Time': [6.5, 8.1, np.nan, 2.3, 75.0, 8.1],
    'Chat_Text': [
        "I was charged twice for my subscription!",
        "My app crashes when I try to open it.",
        "Need help with resetting my password.",
        "Billing page isn't loading on my browser.",
        "Support is very slow. Been waiting forever!",
        "My app crashes when I try to open it."
    ]
}

df_raw = pd.DataFrame(data)

# Show raw data (Before transformation)
print("=== Raw Dataset ===")
print(df_raw)

# -------------------------
# 1. Remove Duplicates
df = df_raw.drop_duplicates()

# 2. Handle Missing Values
df['Chat_Duration'] = df['Chat_Duration'].fillna(df['Chat_Duration'].mean())
df['Response_Time'] = df['Response_Time'].fillna(df['Response_Time'].mean())
df['Issue_Type'] = df['Issue_Type'].fillna(df['Issue_Type'].mode()[0])

# 3. Remove Outliers using IQR for Chat_Duration
Q1 = df['Chat_Duration'].quantile(0.25)
Q3 = df['Chat_Duration'].quantile(0.75)
IQR = Q3 - Q1
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
df = df[(df['Chat_Duration'] >= lower_bound) & (df['Chat_Duration'] <= upper_bound)]

# 4. One-Hot Encoding for Categorical Features
categorical_cols = ['User_Region', 'Issue_Type']
# Changed sparse=False to sparse_output=False for compatibility with newer scikit-learn vers
encoder = OneHotEncoder(sparse_output=False)
encoded = encoder.fit_transform(df[categorical_cols])
encoded_df = pd.DataFrame(encoded, columns=encoder.get_feature_names_out(categorical_cols))

# 5. Scaling Numerical Features
scaler = StandardScaler()
numeric_cols = ['Chat_Duration', 'Response_Time']
scaled = scaler.fit_transform(df[numeric_cols])
scaled_df = pd.DataFrame(scaled, columns=[f"{col}_scaled" for col in numeric_cols])

# 6. Final Combined DataFrame
df_processed = pd.concat([df[['Chat_ID', 'Chat_Text']].reset_index(drop=True),
                          encoded_df.reset_index(drop=True),
                          scaled_df.reset_index(drop=True)], axis=1)
```

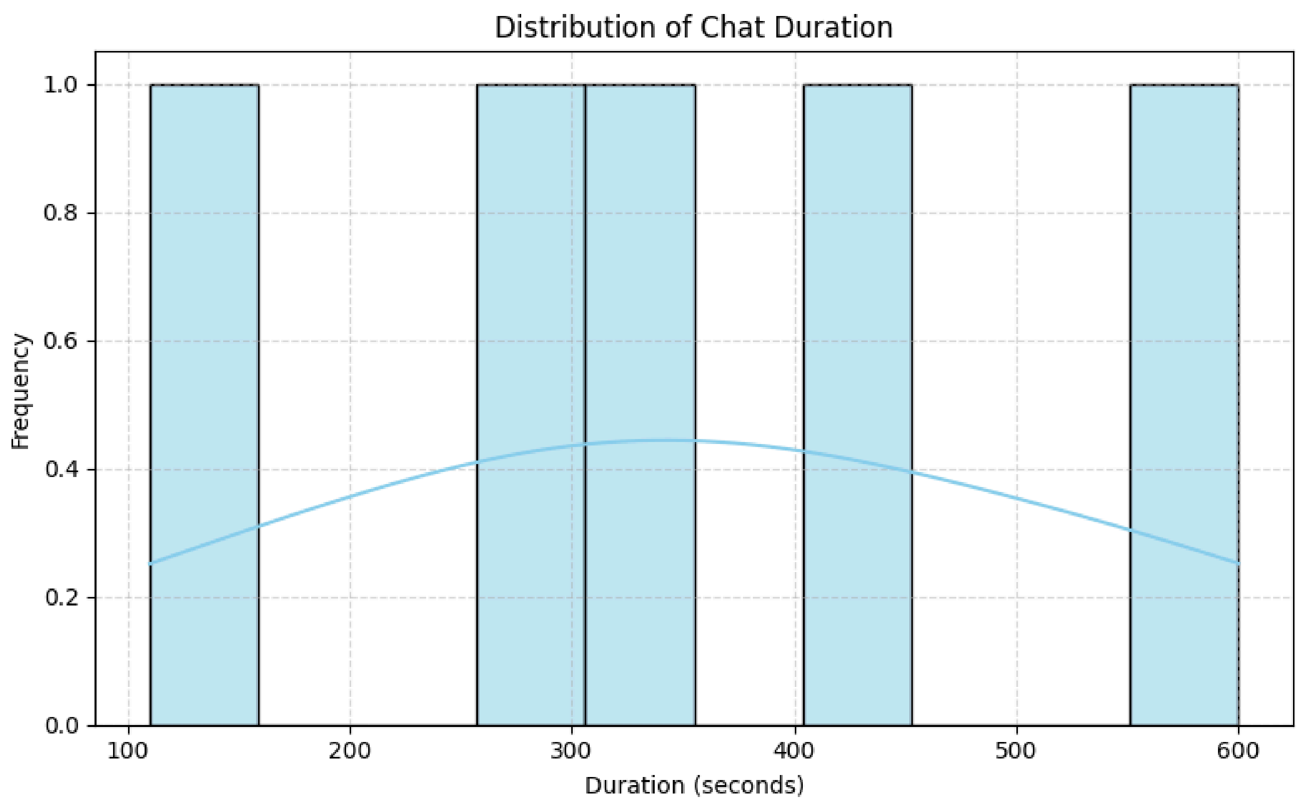
```
# ------------------------
# Show processed data (After transformation)
print("\n=== Processed Dataset ===")
print(df_processed)
```

```
=== Raw Dataset ===
   Chat_ID User_Region Issue_Type  Chat_Duration  Response_Time  \
0     1001        East    Billing          320.0            6.5
1     1002        West  Technical            NaN            8.1
2     1003       South       None         2000.0            NaN
3     1004        East    Billing          295.0            2.3
4     1005       North  Technical          110.0           75.0
5     1002        West  Technical            NaN            8.1

                                  Chat_Text
0     I was charged twice for my subscription!
1        My app crashes when I try to open it.
2        Need help with resetting my password.
3    Billing page isn't loading on my browser.
4  Support is very slow. Been waiting forever!
5        My app crashes when I try to open it.

=== Processed Dataset ===
   Chat_ID                                    Chat_Text  User_Region_East  \
0     1001     I was charged twice for my subscription!               1.0
1     1002        My app crashes when I try to open it.               0.0
2     1004    Billing page isn't loading on my browser.               1.0
3     1005  Support is very slow. Been waiting forever!               0.0

   User_Region_North  User_Region_West  Issue_Type_Billing  \
0                0.0               0.0                 1.0
1                0.0               1.0                 0.0
2                0.0               0.0                 1.0
3                1.0               0.0                 0.0

   Issue_Type_Technical  Chat_Duration_scaled  Response_Time_scaled
0                   0.0             -0.152544             -0.547138
1                   1.0              1.593410             -0.494002
2                   0.0             -0.273372             -0.686621
3                   1.0             -1.167494              1.727760
<ipython-input-6-b752be9532d3>:34: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df['Chat_Duration'] = df['Chat_Duration'].fillna(df['Chat_Duration'].mean())
<ipython-input-6-b752be9532d3>:35: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df['Response_Time'] = df['Response_Time'].fillna(df['Response_Time'].mean())
<ipython-input-6-b752be9532d3>:36: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

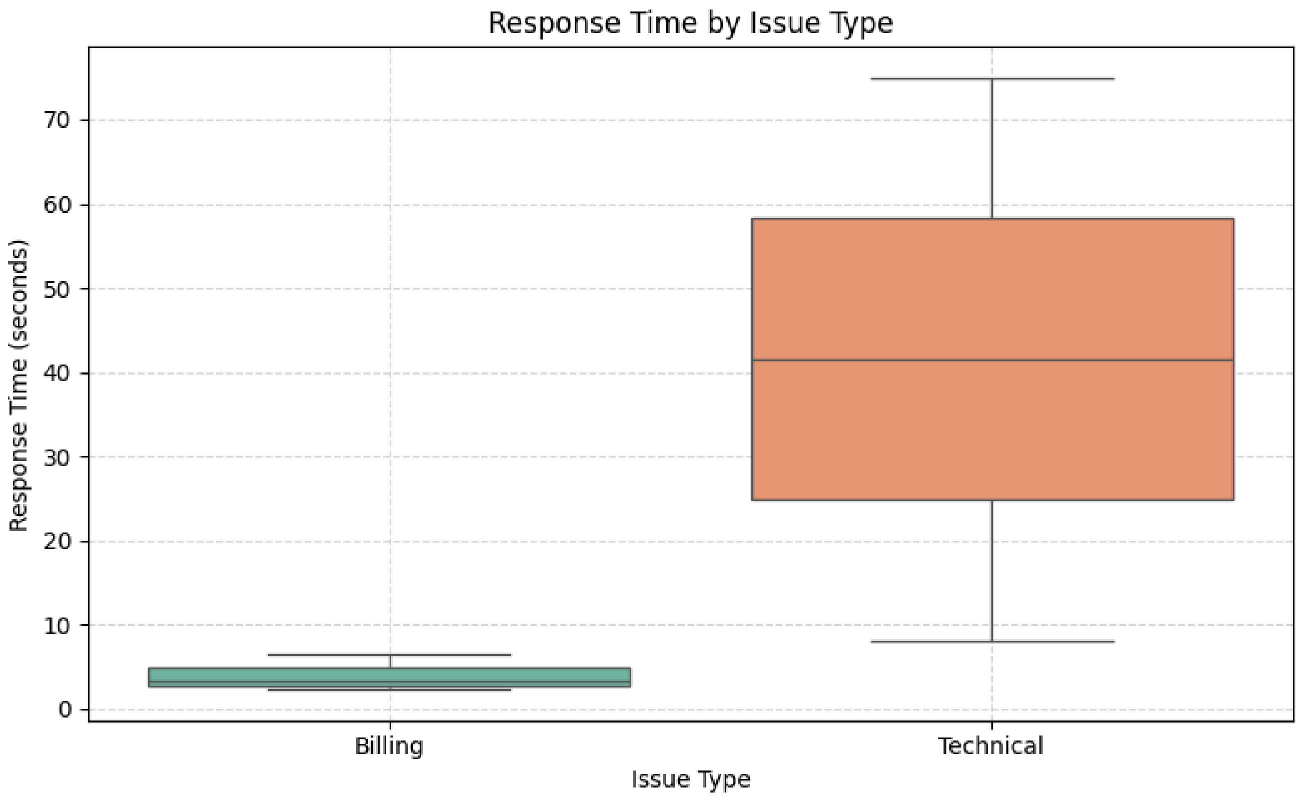
```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  df['Issue_Type'] = df['Issue_Type'].fillna(df['Issue_Type'].mode()[0])
```

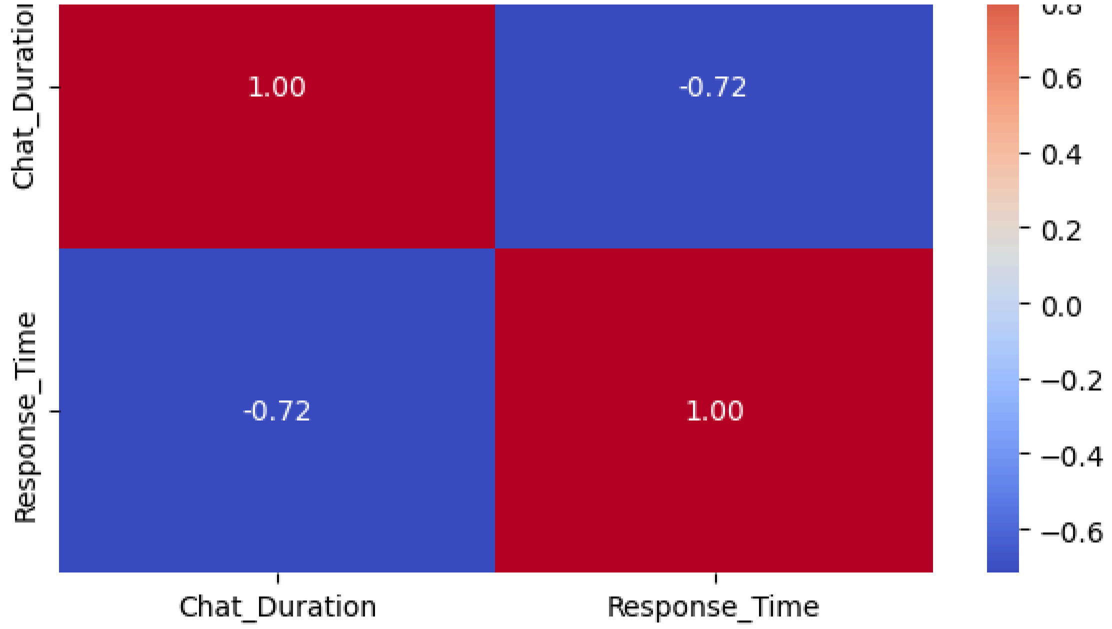
## *Exploratory Data Analysis(EDA) *

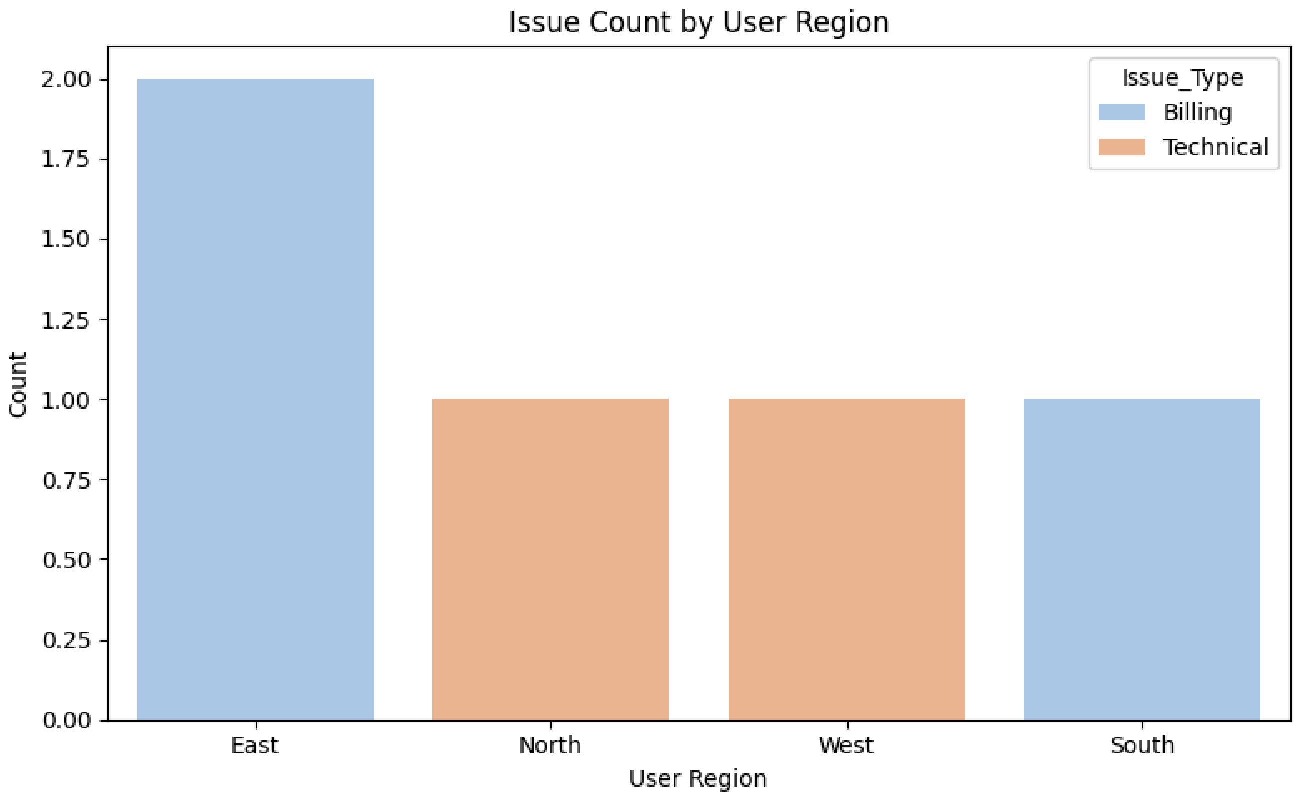
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Sample cleaned dataset
data = {
    'Chat_Duration': [320, 295, 110, 600, 450],
    'Response_Time': [6.5, 2.3, 75.0, 8.1, 3.4],
    'User_Region': ['East', 'East', 'North', 'West', 'South'],
    'Issue_Type': ['Billing', 'Billing', 'Technical', 'Technical', 'Billing']
}
df = pd.DataFrame(data)

# --------- 1. Histogram: Chat Duration ---------
plt.figure(figsize=(8, 5))
sns.histplot(df['Chat_Duration'], bins=10, kde=True, color='skyblue', edgecolor='black')
plt.title('Distribution of Chat Duration')
plt.xlabel('Duration (seconds)')
plt.ylabel('Frequency')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig('histogram_chat_duration.png')  # Screenshot file
plt.show()

# --------- 2. Boxplot: Response Time by Issue Type ---------
plt.figure(figsize=(8, 5))
sns.boxplot(x='Issue_Type', y='Response_Time', data=df, palette='Set2')
plt.title('Response Time by Issue Type')
plt.xlabel('Issue Type')
plt.ylabel('Response Time (seconds)')
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.savefig('boxplot_response_time_issue.png')  # Screenshot file
plt.show()

# --------- 3. Heatmap: Correlation Matrix ---------
corr = df[['Chat_Duration', 'Response_Time']].corr()
plt.figure(figsize=(6, 4))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Feature Correlation Heatmap')
plt.tight_layout()
plt.savefig('heatmap_correlation.png')  # Screenshot file
plt.show()

# --------- 4. Bar Chart: Count of Issues per Region ---------
plt.figure(figsize=(8, 5))
sns.countplot(x='User_Region', hue='Issue_Type', data=df, palette='pastel')
plt.title('Issue Count by User Region')
plt.xlabel('User Region')
plt.ylabel('Count')
plt.tight_layout()
plt.savefig('barchart_region_issue.png')  # Screenshot file
plt.show()
```



```
<ipython-input-8-001ef87dcd0e>:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0

  sns.boxplot(x='Issue_Type', y='Response_Time', data=df, palette='Set2')
```




Chat_Duration
Response_Time
1.00
-0.72
-0.72
1.00
Chat_Duration
Response_Time
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


Issue Count by User Region
Issue_Type
Billing
Technical
Count
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
East
North
West
South
User Region

feature engineering

```
import pandas as pd
import numpy as np

# Example raw dataset for customer support interactions
data = {
    'Chat_ID': [1001, 1002, 1003, 1004, 1005],
    'Chat_Timestamp': pd.to_datetime([
        '2025-04-20 10:15:00',
        '2025-04-20 11:30:00',
        '2025-04-21 09:45:00',
        '2025-04-21 14:00:00',
        '2025-04-22 16:20:00'
    ]),
    'User_Region': ['East', 'West', 'South', 'East', 'North'],
    'Issue_Type': ['Billing', 'Technical', 'Billing', 'Technical', 'Technical'],
    'Chat_Duration': [320, np.nan, 2000, 295, 110],          # in seconds; contains a missir
    'Response_Time': [6.5, 8.1, np.nan, 2.3, 75.0],            # in seconds; contains a miss
    'Message_Count': [15, 10, 25, 8, 30],                      # number of messages exchange
    'Chat_Text': [
        "I was charged twice for my subscription!",
        "My app crashes when I try to open it.",
        "Need help with resetting my password.",
        "Billing page isn't loading on my browser.",
        "Support is very slow. Been waiting forever!"
    ]
}

raw_df = pd.DataFrame(data)

# Display the raw dataset
print("=== Raw Dataset Sample ===")
print(raw_df)
```

```
=== Raw Dataset Sample ===
   Chat_ID      Chat_Timestamp User_Region Issue_Type  Chat_Duration  \
0     1001 2025-04-20 10:15:00        East    Billing          320.0
1     1002 2025-04-20 11:30:00        West  Technical            NaN
2     1003 2025-04-21 09:45:00       South    Billing         2000.0
3     1004 2025-04-21 14:00:00        East  Technical          295.0
4     1005 2025-04-22 16:20:00       North  Technical          110.0

   Response_Time  Message_Count                                    Chat_Text
0            6.5             15     I was charged twice for my subscription!
1            8.1             10        My app crashes when I try to open it.
2            NaN             25        Need help with resetting my password.
3            2.3              8    Billing page isn't loading on my browser.
4           75.0             30  Support is very slow. Been waiting forever!
```

## Model Building step1: (including typical chatbot interaction features)

```python
import pandas as pd
import numpy as np

# Create sample raw dataset
data = {
    'Chat_ID': range(1001, 1011),
    'Chat_Timestamp': pd.to_datetime([
        '2025-05-10 10:15:32', '2025-05-10 11:05:12', '2025-05-11 09:45:00',
        '2025-05-11 14:00:10', '2025-05-11 16:20:00', '2025-05-12 08:40:50',
        '2025-05-12 10:20:10', '2025-05-12 11:50:30', '2025-05-12 13:15:00',
        '2025-05-12 15:45:15'
    ]),
    'User_Region': ['East', 'West', 'South', 'East', 'North',
                    'East', 'South', 'West', 'North', 'East'],
    'Issue_Type': ['Billing', 'Technical', 'Billing', 'Technical', 'Technical',
                   'General', 'Billing', 'Technical', 'General', 'Technical'],
    'Chat_Duration': [320, np.nan, 2000, 295, 110, 450, 780, 560, 310, 640],  # seconds
    'Response_Time': [6.5, 8.1, np.nan, 2.3, 75.0, 5.2, 9.5, 6.0, 3.8, 7.0],  # seconds
    'Message_Count': [15, 10, 25, 8, 30, 12, 20, 22, 18, 28],
    'Chat_Text': [
        "I was charged twice for my subscription.",
        "App crashes when I open it.",
        "Need help resetting password.",
        "Billing page not loading in browser.",
        "Support is slow, waited too long.",
        "How do I update my plan?",
        "My invoice has the wrong address.",
        "Still facing login errors despite resetting.",
        "Where can I find my usage history?",
        "Help! Website keeps logging me out randomly."
    ],
    'Satisfaction_Rating': [4, 2, 3, 1, 1, 5, 4, 2, 5, 2],
    'Escalated': [0, 1, 0, 1, 1, 0, 0, 1, 0, 1]
}

df = pd.DataFrame(data)

# Save to CSV (optional)
df.to_csv("chatbot_raw_data.csv", index=False)

# Preview
print(df.head())
```

```
   Chat_ID      Chat_Timestamp User_Region Issue_Type  Chat_Duration  \
0     1001 2025-05-10 10:15:32        East    Billing          320.0
1     1002 2025-05-10 11:05:12        West  Technical            NaN
2     1003 2025-05-11 09:45:00       South    Billing         2000.0
3     1004 2025-05-11 14:00:10        East  Technical          295.0
4     1005 2025-05-11 16:20:00       North  Technical          110.0

   Response_Time  Message_Count                                 Chat_Text  \
0            6.5             15  I was charged twice for my subscription.
1            8.1             10                 App crashes when I open it.
2            NaN             25             Need help resetting password.
3            2.3              8      Billing page not loading in browser.
4           75.0             30         Support is slow, waited too long.

   Satisfaction_Rating  Escalated
0                    4          0
1                    2          1
2                    3          0
3                    1          1
4                    1          1
```

## Model Building step:(Baseline & Advanced)

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
from sklearn.metrics import classification_report
from sklearn.pipeline import Pipeline
import numpy as np

# Prepare features
features = ['User_Region', 'Issue_Type', 'Chat_Duration', 'Response_Time', 'Message_Count']
X = df[features]
y = df['Escalated']

# Handle missing values
X['Chat_Duration'] = X['Chat_Duration'].fillna(X['Chat_Duration'].median())
X['Response_Time'] = X['Response_Time'].fillna(X['Response_Time'].median())

# One-hot encode categorical features
X = pd.get_dummies(X)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    'Random Forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'XGBoost': XGBClassifier(use_label_encoder=False, eval_metric='logloss', random_state=42
}

# Train and evaluate each model
for name, model in models.items():
    print(f"\n🔷 Training: {name}")
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    print("Classification Report:")
    print(classification_report(y_test, y_pred))
```

```
<ipython-input-11-78deee66cb84>:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  X['Chat_Duration'] = X['Chat_Duration'].fillna(X['Chat_Duration'].median())
<ipython-input-11-78deee66cb84>:17: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_
  X['Response_Time'] = X['Response_Time'].fillna(X['Response_Time'].median())

🔷 Training: Logistic Regression
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       1.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0


🔷 Training: Random Forest
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2


🔷 Training: XGBoost
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2

/usr/local/lib/python3.11/dist-packages/xgboost/core.py:158: UserWarning: [05:43:12] WAF
Parameters: { "use_label_encoder" } are not used.

  warnings.warn(smsg, UserWarning)
```

import required libraries in Model Evalution

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import (
    accuracy_score, f1_score, roc_auc_score, confusion_matrix,
    roc_curve, classification_report, mean_squared_error
)
```

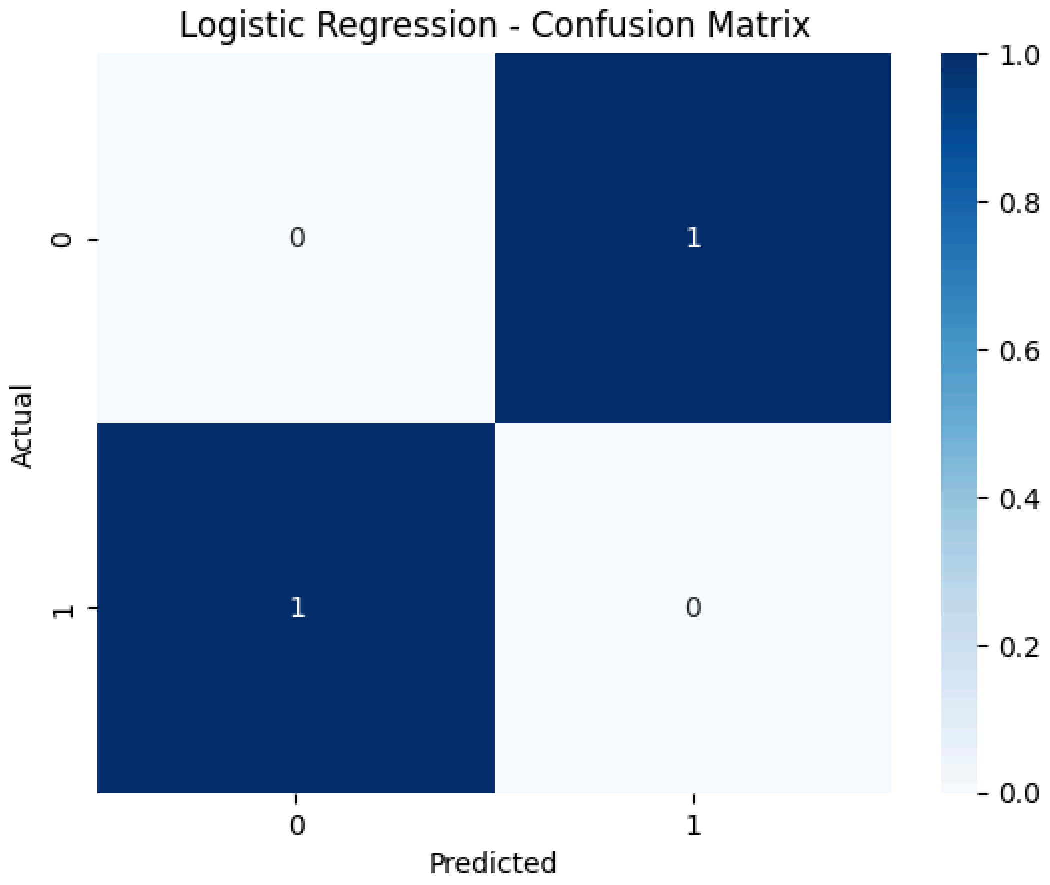
## Define Evaluation Matrices

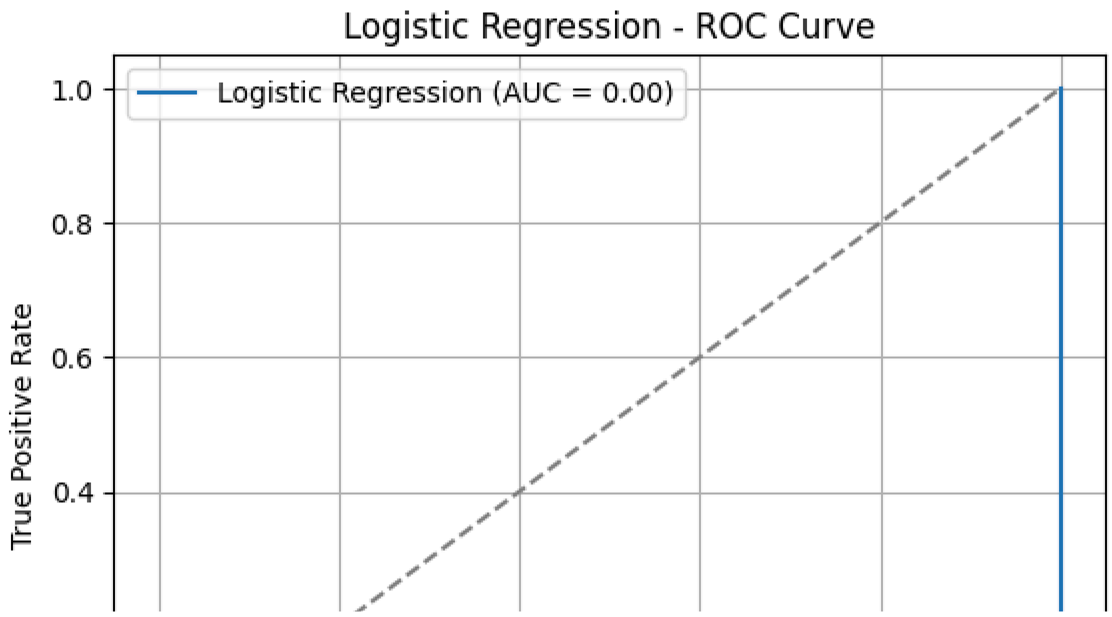
```
def evaluate_model(name, model, X_test, y_test):
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test)[:, 1]

    accuracy = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    roc = roc_auc_score(y_test, y_proba)
    rmse = np.sqrt(mean_squared_error(y_test, y_proba))

    print(f"\n📌 Evaluation for: {name}")
    print("Classification Report:")
    print(classification_report(y_test, y_pred))

    # Confusion Matrix
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'{name} - Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

    # ROC Curve
    fpr, tpr, _ = roc_curve(y_test, y_proba)
    plt.plot(fpr, tpr, label=f'{name} (AUC = {roc:.2f})')
    plt.plot([0, 1], [0, 1], linestyle='--', color='gray')
    plt.title(f'{name} - ROC Curve')
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.legend()
    plt.grid()
    plt.show()

    return {
        'Model': name,
        'Accuracy': accuracy,
        'F1 Score': f1,
        'ROC AUC': roc,
        'RMSE': rmse
    }
```

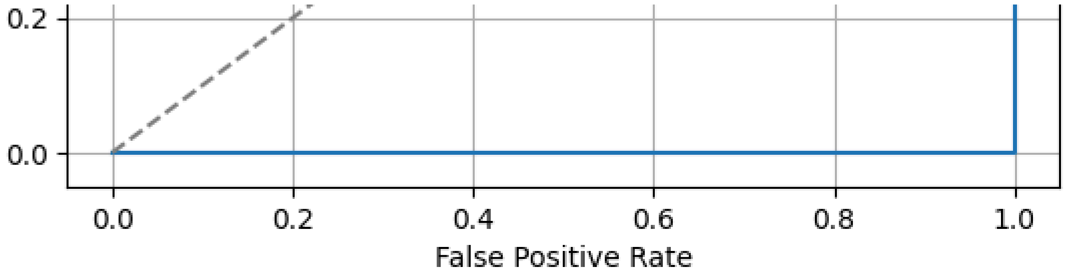
## Run Evaluation for Multiple Models

```
# Example:
results = []
# Access models from the 'models' dictionary
results.append(evaluate_model("Logistic Regression", models['Logistic Regression'], X_test,
results.append(evaluate_model("Random Forest", models['Random Forest'], X_test, y_test))
results.append(evaluate_model("XGBoost", models['XGBoost'], X_test, y_test))

# Model Comparison Table
results_df = pd.DataFrame(results)
print("\n📋 Model Comparison Table:")
print(results_df)

# Plotting comparison
results_df.set_index('Model')[['Accuracy', 'F1 Score', 'ROC AUC']].plot(kind='bar', figsize=
plt.title('Model Evaluation Metrics')
plt.ylabel('Score')
plt.ylim(0, 1)
plt.grid(axis='y')
plt.tight_layout()
plt.show()
```

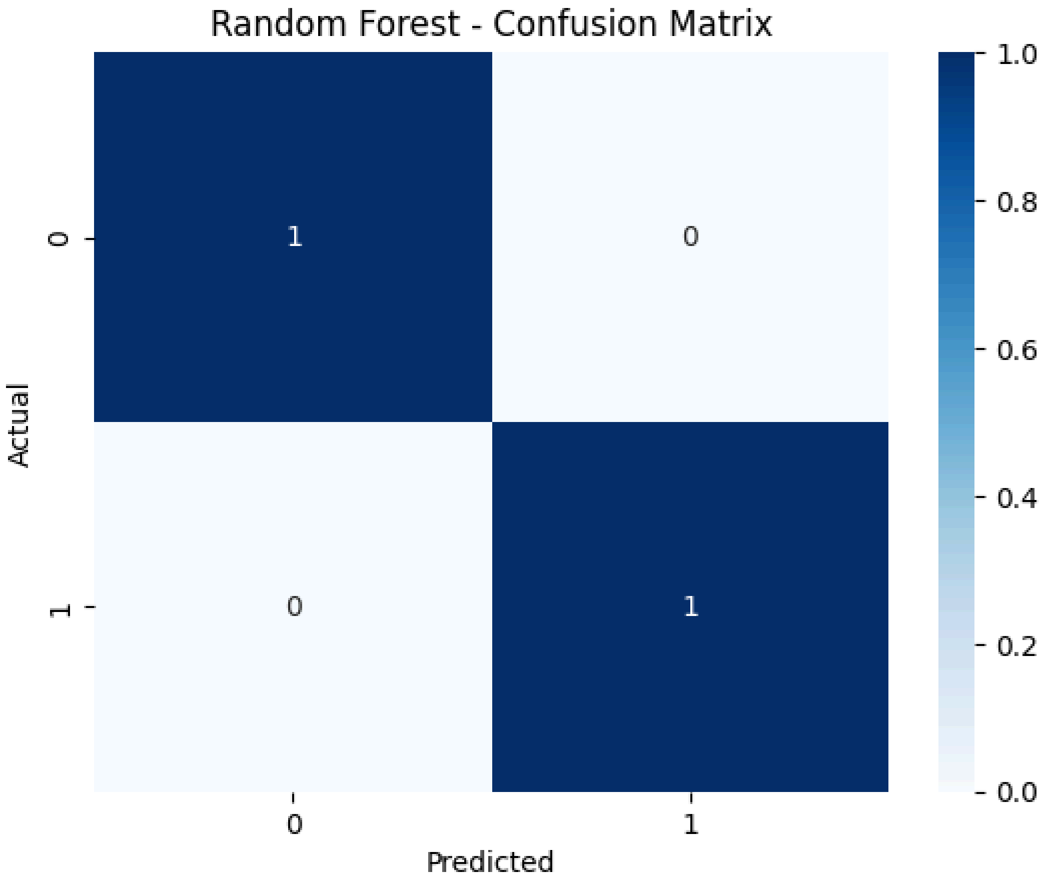
```
📌 Evaluation for: Logistic Regression
Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       1.0
           1       0.00      0.00      0.00       1.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0
```

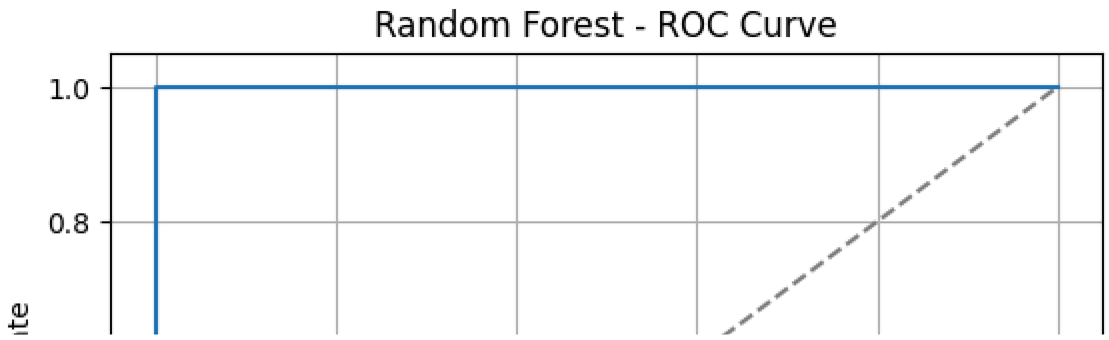


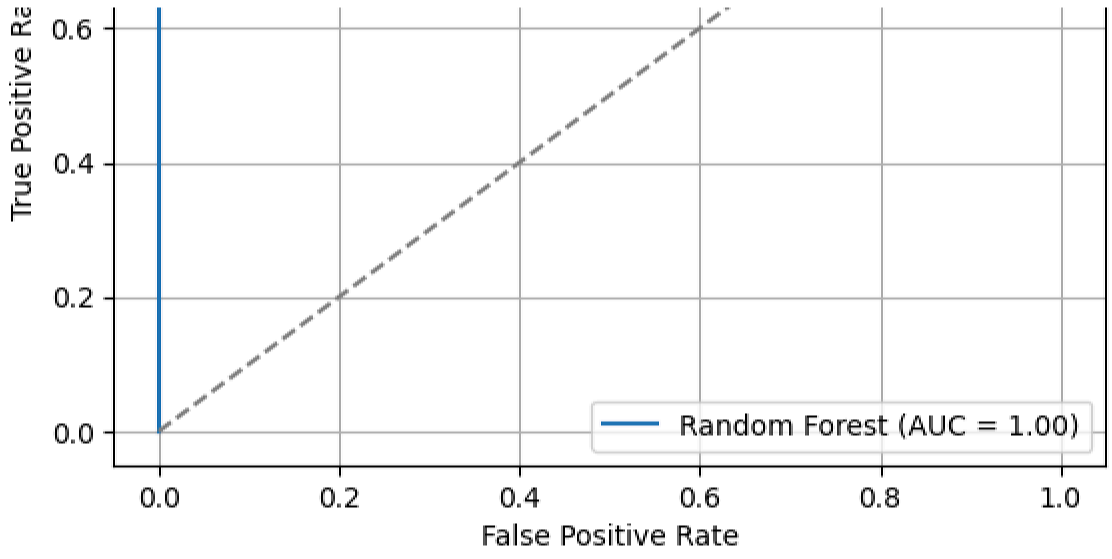




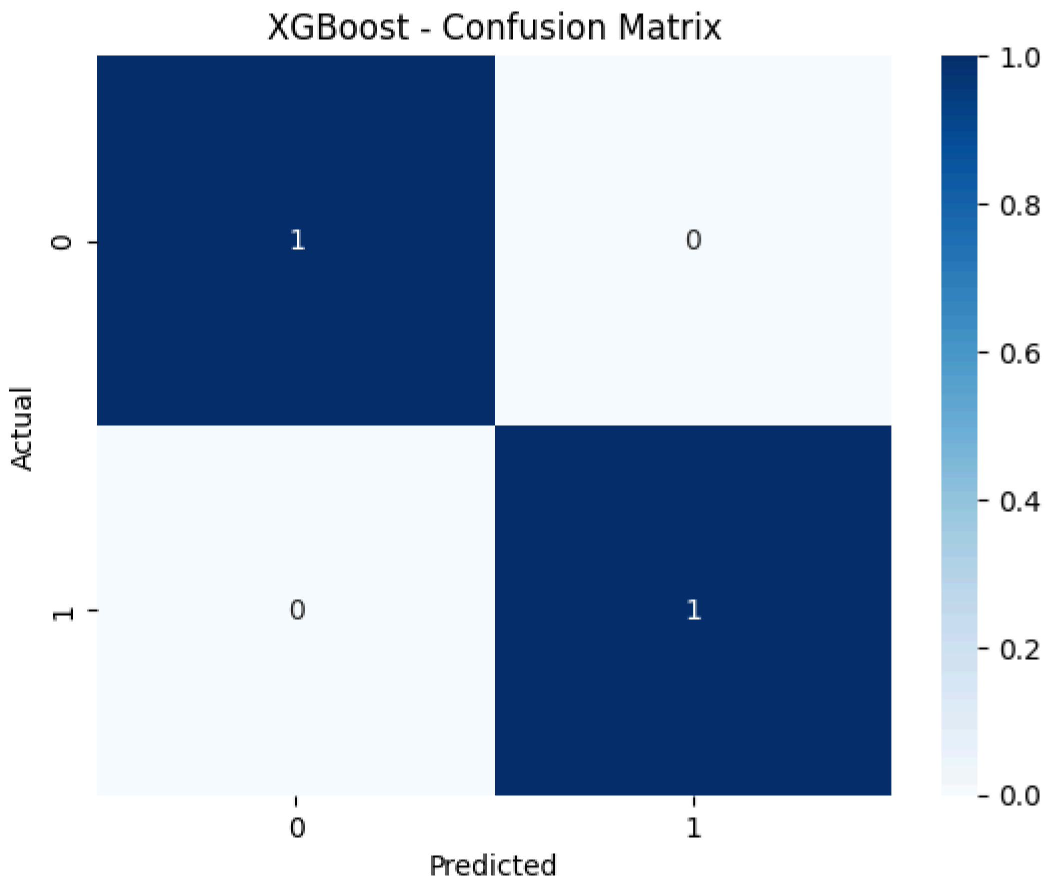
```
📌 Evaluation for: Random Forest
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```



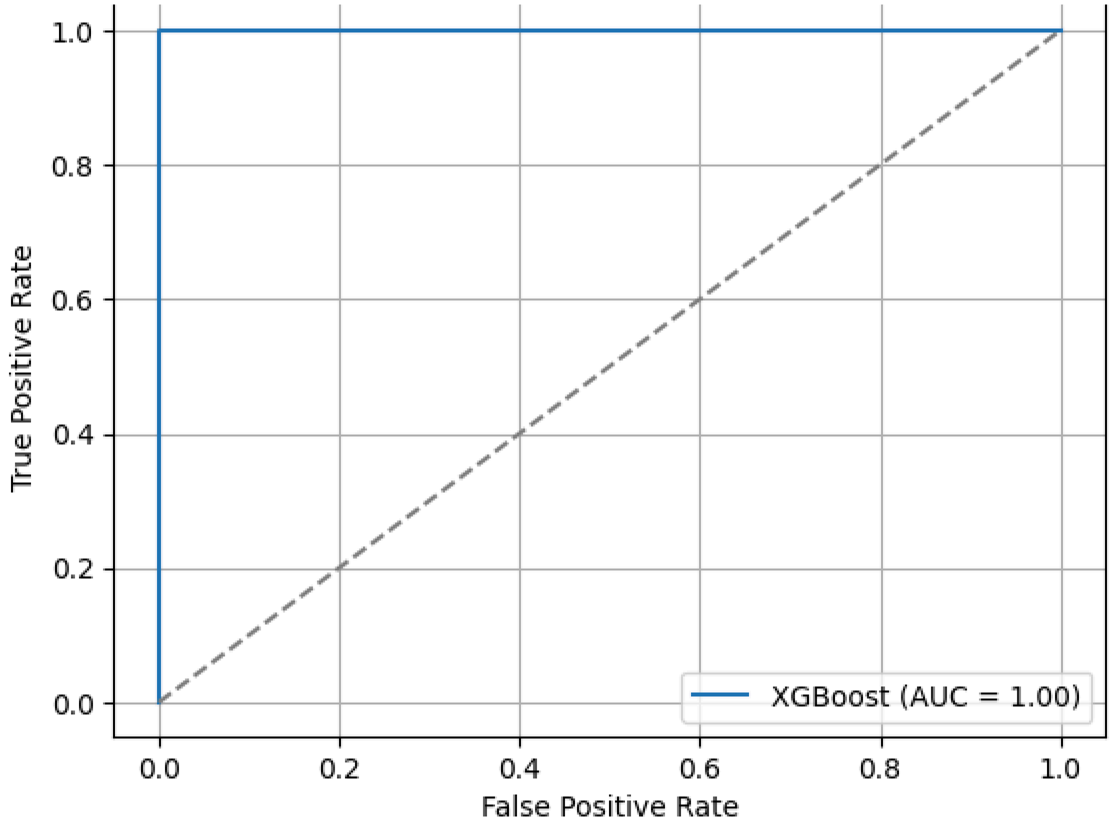




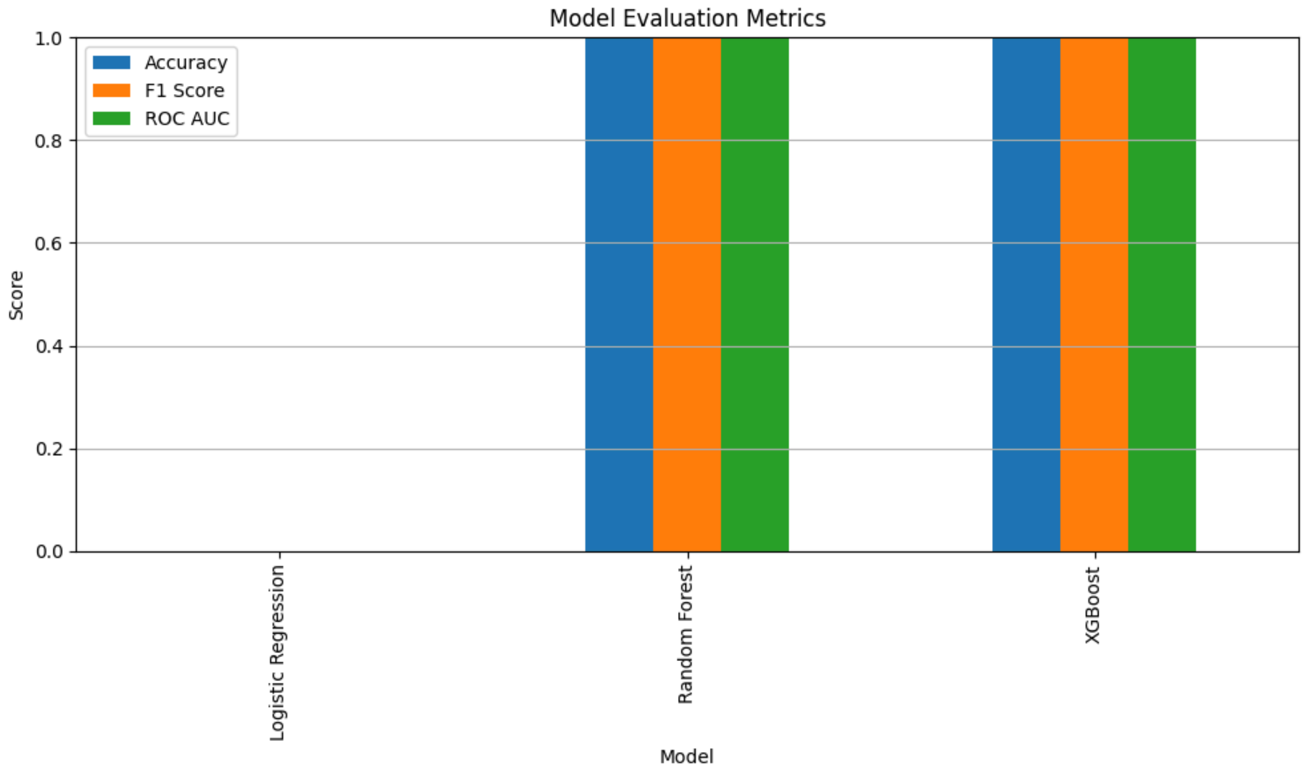
```
📌 Evaluation for: XGBoost
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```



XGBoost - ROC Curve



```
📋 Model Comparison Table:
                 Model  Accuracy  F1 Score  ROC AUC      RMSE
0  Logistic Regression       0.0       0.0      0.0  0.728259
1        Random Forest       1.0       1.0      1.0  0.270740
2              XGBoost       1.0       1.0      1.0  0.425557
```


Model Evaluation Metrics
Accuracy
F1 Score
ROC AUC
Score
Model
Logistic Regression
Random Forest
XGBoost